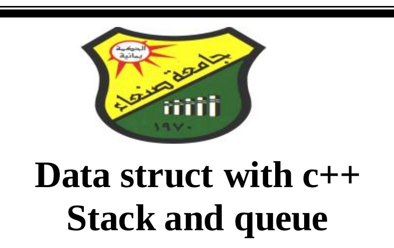






**ما هو المكدس Stack؟؟**

**هو احد هياكل البيانات ، الاكثر سهولة ، يستخدم بكثرة في عالم الحواسيب : مترجمات ، الات حاسبة ، المعالجات و...الخ.**

**ومفهومها بسيط جدا ،، ادرس المثال الاتي:**

**تخيل ان لدينا مجموعة من الاطباق "الصحون" ، مرصوصة فوق بعضها ،اي صحن وفوقه واحد اخر الى ان نصل الى اخر صحن.**

**الان لكي نضيف صحن اخر الى المجموعة ، يجب ان نضعه على رأسهم ، يعني اعلى شيء top**

**واذا اردنا ان نسحب اي عنصر ، يجب ان نسحب الذي فوقه اولا.**

**اي لا تستطيع سحب الصحن الرابع مثلا دون ان نسحب الصحون التي تقع فوقه ، والا ستنكسر الصحون.**

**مثلا اذا اردنا سحب الصحن الثاني ، يجب ان نسحب كل الصحون التي تقع فوقها ، حتى نصل الى الصحن الثاني و نسحبه ،، وتكون المجموعة الجديدة مكونة من الصحن الاول فقط ، وذلك لاننا سحبنا كل الصحون التي تقع فوقها.**

**ويتضح المثال بهذا التعريف:**

**المكدس هو عبارة عن فكرة "طريقة" تطبق على المصفوفة "ليس في كل الحالات ، ولكن سنستخدم المصفوفة هنا" ،، بحيث ان ادخال العناصر يتم من اعلى "كما في حالة الصحون" ، وكذلك سحب العناصر يتم من اعلى.**

**وذلك على خلاف المصفوفة العادية ، مثلا اذا ادخلنا في اي مصفوفة العناصر**

**2ثم ٤ ، ٦ ، ٨ ، ١٠.**

**واردنا عرضها على الشاشة فان النتيجة هي:**

**2 ، 4 ، 6 ، 8 ، 10اي على نفس ترتيب الادخال.**

ولكن اذا ادخلنا الاعداد السابقة في مكدس ، وعرضنا عناصر المكدس على الشاشة ، فالنتيجة هي:

**10 ، 8 ، 6 ، 4 ، 2اي على عكس ترتيب الادخال.**

اي ان العنصر الداخل الى المكدس اولا ، هو الذي سيخرج في الاخير

والعنصر الداخل الى اعلى المكدس ، هو الذي سيخرج اولا.

**Last in First Out**

لذلك تجد دائما مع المكدس هذه العباره ، وتختصر ب **LIFO**

خلاصة :

*المكدس هو عبارة عن هيكل بيانات.

*يتم تطبيق المكدس من خلال المصفوفات او من خلال القوائم المرتبطة "سنتحدث عنها لاحقا"

*يتبع المكدس مفهوم **LIFO** ، اي العنصر الذي يدخل في الاخير ، هو الذي يخرج اولا.

*لادخال عنصرفي المكدس يجب ان نضعه فوق اعلى عنصر.

كيف سنعرف ان العنصر هو اعلى ام لا ؟

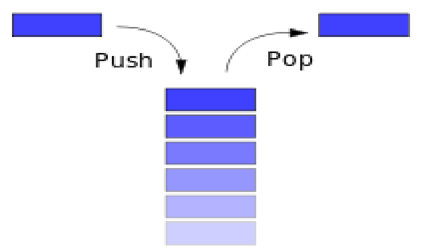
سنحتاج الى مؤشر للمصفوفة "عدد صحيح " ، وذلك لكي نعرف من هو اعلى عنصر ،، وليكن اسمه.**top**

في الحقيقة **top** ليس مؤشر **pointer** وانما هو عدد **int** ، ولكن نستخدمه كدليل الى العنصر الاعلى في المصفوفة ، يعني اذا كان عندي مصفوفة من ١٠ عناصر ، والمستخدم ادخل قيمة اول عنصرين ، فان المتغير **top** سيحمل القيمة ١ ، دلالة على ان العنصر الثاني هو اعلى عنصر.

*عملية ادخال العنصر الى المكدس ، تسمى **push** ، والتي تعني دفع العنصر الى اعلى المكدس.

*عملية سحب العنصر من المكدس ، تسمى **pop**

صورة توضيحية:




اعتقد ان المفهوم اتضح قليلا ،، والكود هو خير الكلام ،، ما " قل و دل"



اسم الطالب :- عبدا لرحمن يحي صلح
جامعة صنعاء /كلية الحاسوب وتكنولوجيا المعلومات

## العمليات على المكدس

**two operations: حتى الان لدينا عملتين وهم**
**عملية ادخال العناصر الى المكدس وسميناها push**
**عملية اخراج العناصر من المكدس وسميناها pop**

**العملية push تقوم بادخال العنصر الى المكدس ،، هذا العنصر سيقوم المستخدم بتمريره الى الدالةpush()**
**اي اذا اردنا ادخال العنصر ٧ مثلا الى المكدس ، كل ما علينا هو استدعاء الدالة وتمرير العنصر ٧ اليها كالاتي:**

**كود:**

```
push(7);
```

**العملية الاخرى وهي pop تقوم بسحب العناصر من المكدس ،، ولتنفيذ هذه العملية يجب ان تستدعي الدالة**
**pop()ووضع متغير لكي يحمل قيمة العنصر الذي تم سحبه من المكدس، بهذا الشكل:**
**كود:**

```
int var=pop();
```

**خوارزمية الادخال والاخراج في stack**

ps ←

2. push the data:- ادخال البيانات للـStack
Input: Top
e // item of char type
output: item e will be placed at top position.
push algorithm
Begin: IF Stack is not full then
Top < size of the stack
Begin:
increment Top by one
placed item e at position Top.
End
Else Display ("stack is full")
End

3. pop algorithme حذف عناصر من الـstack
ويتم الحذف من خلال حذف الموقع المؤشر اليه بـTop
Input: Top
output:- e
pop algorithm
Begin:- If stack is not empty then
Top > 0
Begin:
e = item at position Top.
decrement top by one
End
Else
Display ('empty').
End



اسم الطالب :- عبدا لرحمن يحي صلح
جامعة صنعاء /كلية الحاسوب وتكنولوجيا المعلومات

## Stack اولاً أمثلة على :

Push Pop

الداخل اولاً الخارج اخيراً(الداخل اخرُ الخارج اولاً)

برنامج يقوم بعملية عملية الادخال والاخراج :

```
#include<iostream.h>
#include<conio.h>
int size=10;//البرنامج على عالمي كمتغير الحجم عرفناء هناء
int a[10],top=-1;//عرفناء مصفوفة حجمها10وهيئناء التوب با -1
int pop();//عرفناء دالة الاخراج
void push(int[],int);//عرفناء دالة وعرفناء موقعين داخلهاء
main()
{
int i,k;//عرفناء متغيرين لغرض استخدامها في البرنامج
for(i=0;i<size;i++)//هناء يتم عمل دوارة ويستخدم كعداد للمرور بالشرط
{
cout<<"Enter the item push pleas\n";
cin>>k;//هناءيتم ادخال البينات
push(a,k);//هناء يتم تضمين العناصر المدخلة الى دالة البوش
}
for(i=0;i<size;i++)//هناء يتم عمل دوارة ويستخدم كعداد للمرور بالشرط
cout<<"\nThe elemant pop\n"<<pop()<<" \n";//هناء يتم استدعاء دالة البوب
getch();}//تقوم بعمل ايقاف الشاشة
void push(int a[],int k)//هاذة دالة البوش
{
if(top==size-1)//هاذا شرط اذاكان التوب يساوي الحجم-1 فان الستاك مليانة
cout<<" FULL STACK";
else//اذا الستاك لم تمتلى فيتم
a[++top]=k;//هناء يتم زيادة التوب بواحد
}
int pop(){//البوش دالة
if(top<0)//هناء يتم اذا كان التوب اصغر من الصفر
cout<<"EMPTE STACK";//فن الدالة خالية
else//مالم يتم
return a[top--];//نقصان ال توب بواحد
}

//**************************************************************************
```

## برنامج يقوم بإضافة ستاك فارغة ثم يضيف اليها ٣ عناصر .. بعد ذألك يقوم البرنامج بحذف الرأس وعرض بقية العناصر على الشاشة:

```
#include <iostream>
using namespace std;
struct Node
{
int Data;
Node *Next;
```

```
};
Node* InitStack(Node *S)
{
S = NULL;
return S;
}
bool IsEmpty(Node *S)
{
if (S == NULL)
return true;
else
return false;
}
Node* Push(int x, Node *S)
{
Node *P = new(Node);
P -> Data = x;
P -> Next = S;
S = P;
return S;
}
int Top(Node *S)
{
if (!IsEmpty(S))
return S->Data;
else
     cout<<"Stack empty ..."; // un derflow
}
Node* Pop(Node *S)
{
if (!IsEmpty(S))
{
Node *P = S;
S = S ->Next;
delete P;
return S;
}
else
cout<<"Stack empty ...";
}
void main()
{
Node *S;
S = InitStack(S);
```

```
S = Push(10, S);
S = Push(5,S);
S = Push(2,S);
S = Pop(S);
while (!IsEmpty(S))
{
cout<<Top(S)<<"\t";
S=Pop(S);
}
cout<<endl;
}
```

```
//************************************************************
```

**برنامج يقوم بعملية الادخال والاخراج وعملية العرض:**

```
#include<iostream.h>
#include<conio.h>
//----------------------------
int top=0;
   const int size=5;
  int stack[size];
  //--------------------------
  int empty()
  {    if(top==0)
       return 1;
       else
       return 0;
   }
  //----------------------------
  int full()
  {   if(top==size)
       return 1;
       else
       return 0;
  }
  //--------------------------------
 void push(int x)
 {   if(!full())
        stack[top++]=x;
     else
       cout<<"the stack is full\n";
  }
  //------------------------------------
  int pop()
  {   if(!empty())
        return stack[top--];
      else
         cout<<"the stack is empty\n";
  }
  //--------------------------------------
  void show()
```

```
  {if(!empty())
  for(int i=0;i<top;i++)
        cout<<stack[i]<<" ";
  }
  //-----------------------------------
   main()
  {
     int op;
    do
     {
        cout<<"1-add\n";
        cout<<"2-pop\n";
          cout<<"3-show\n";
          cin>>op;
          switch(op)
          { case 1: int x;
                  cout<<"input val\n";
                  cin>>x;
                  push(x);
                  break;
            case 3 : show();
            break;
           case 2: int y =pop();
                              cout<<"\nThe delete of stack\n"<<y;
                   break;
}

 }while(op !=0);//end while--------
     getch();
}//The End-----
```

//*************************************************

## برنامج يقوم بعملية الادخال والاخراج استخدمنا في هاذا المثال struct:

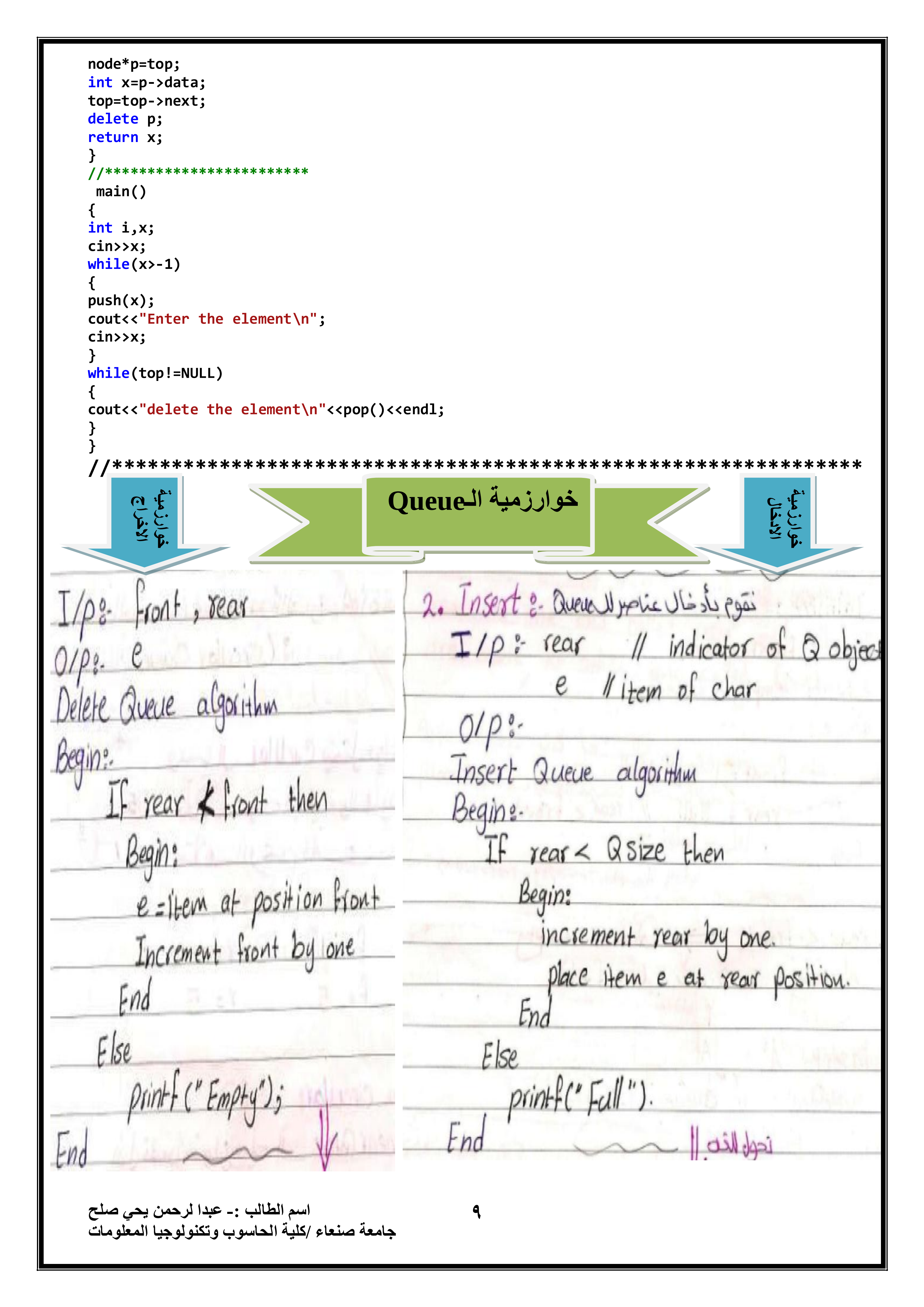
```
#include<iostream.h>
//******************************
const int size=5;
//******************************
struct stack
{int top;
int item[size];
}ps;
//****************************
void initial(struct stack *ps)
{ps->top=-1;
cout<<ps->top;
}
//*****************************
void push(struct stack *ps,int m)
{if(ps->top<(size-1))
{(ps->top)++;
ps->item[ps->top]=m;}
else
      cout<<"full\n";}
//*********************************
void pop(struct stack *ps)
```

```
{if(ps->top>(-1))
{cout<<ps->item[ps->top]<<endl;
(ps->top)--;}
else
      cout<<"empty\n";
}
//*******************************
      int main()
      {
      int n, op;
      initial(&ps);
      do{
                  cout<<"chose 1 to push and 2 to pop\n";
                  cin>>op;
                  switch(op)
                  {case 1:;
                  {cin>>n;
                  push(&ps,n);
                  }
                  break;
                  case 2:

                  pop(&ps);
                  break;
                  default:
                  cout<<"error";
                  }}while(op!=3);
 return 0;  }
```

## هاذا البرنامج باستخدام الـــ Dynamic Stack:

```
#include<iostream.h>
//Dynamic Stack هاذا البرنامج باستخدام الـ
//*******************
struct node
{
int data;
node*next;
};
node*top;
//****************
void push(int c)
{
if(top==NULL)
{
node*p=new node;
p->data=c;
p->next=NULL;
top=p;
}
else
      {
node*p=new node;
p->data=c;
p->next=top;
top=p;
      }  }
//********************
int pop()
{
```



```
node*p=top;
int x=p->data;
top=top->next;
delete p;
return x;
}
//************************
 main()
{
int i,x;
cin>>x;
while(x>-1)
{
push(x);
cout<<"Enter the element\n";
cin>>x;
}
while(top!=NULL)
{
cout<<"delete the element\n"<<pop()<<endl;
}
}
//**************************************************************
```

## خوارزمية الـQueue

خوارزمية الادخال

2. Insert :- نقوم بأدخال عناصر للQueue

I/p :- rear // indicator of Q object
e // item of char

O/p :-

Insert Queue algorithm

Begin:-
    If rear < Qsize then
        Begin:
            increment rear by one.
            place item e at rear position.
        End
    Else
        printf(" Full").
End

تحول الثه ||

خوارزمية الاخراج

I/p :- front , rear

O/p :- e

Delete Queue algorithm

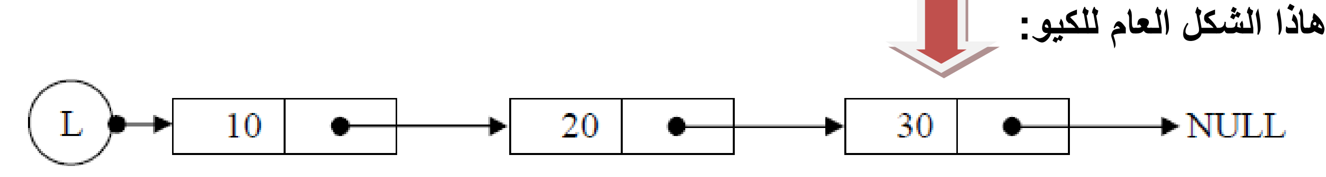
Begin:-
    If rear ≮ front then
        Begin:
            e := item at position front
            Increment front by one
        End
    Else
        printf(" Empty");
End

# ثانياً: Queue :

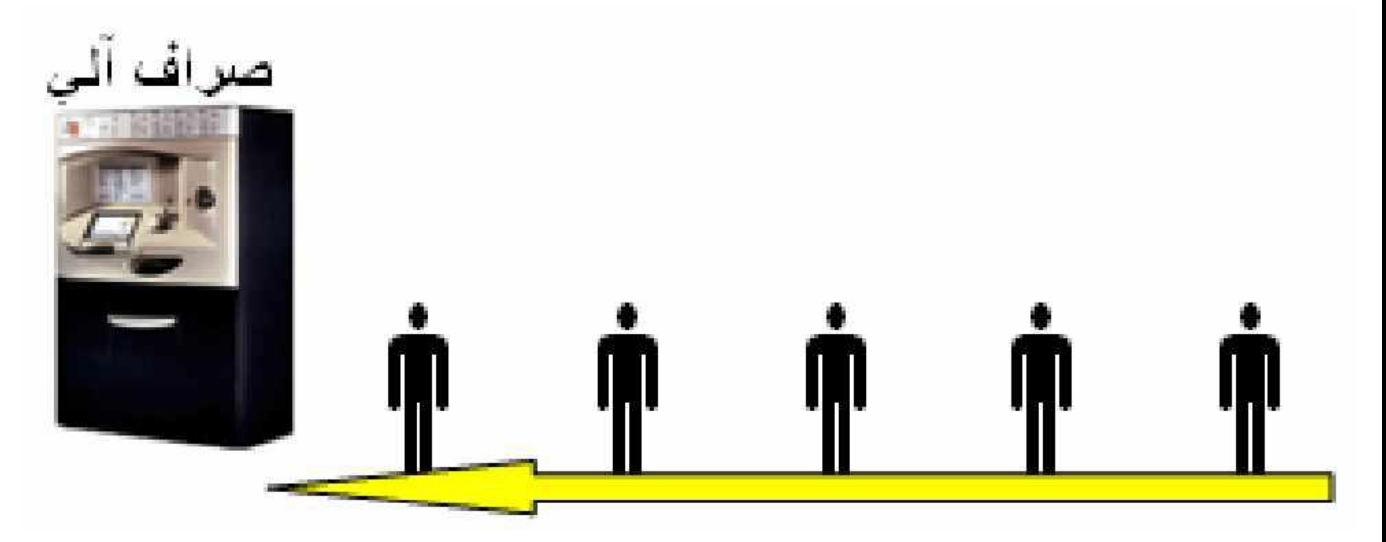
**ما هو : Queue**

**هو عبارة عن هيكل بيانات يستخدم لتخزين البيانات من اجل معالجتها بحيث ان العنصر الذي يدخل اولا للطابور يتم معلجته اولا والعنصر الذي يدخل في الاخر هو الذي يتم المعالجة له في الاخر .**

**مثال على الطابور او الكيو :-**
**خلينا نؤخذ مثال طابور التأشيرة في المطار مثلاً في فترة الصبح لما يكون فيها زحمة المسافرين ولنفرض انو واحد من موظفين المطار يأتي متأخر على المطار ولما دخلها تفجأ في الزحمة الموجودة هناك فحاول انو يتجاوز المسافرين لحتي يدخل على مكانه بس للأسف ما قدر يدخل عارفين ليش ؟؟؟ لانو في قانون في الطابور انو اي عنصر بيجي راح يصف في اخر الطابور وعملية الاضافة على الطابور بنسميها "اينكيو" وبعد فترة تحرك الطابور للامام لانو طلع واحد من المسافرين وحاسب فهي العملية وهي عملية الحذف نسميها "ديكيو" ومتل هاذا لما ياتي دور الموظف .**
**هاذا الشكل العام للكيو:**




**او مثل طابور استخدام الصراف الالي:**




**في المثال السابق فإنه المبدا الذي اتبعنا انه العنصر الذي يدخل في الاول يخرج في الاول وتسمى "فيرست اين فيرست اوت"**

**( first in first out)**

**وبمعنى اخر ان عملية الاضافة تتم على العنصر الاخير والذي سوف نسمية في "رير "والحذف يتم على العنصر الاول والذي سوف نسمية "فرونت " .**

**ملاحظة :-*****

**front:- هو فرونت وهو العنصر الاول الذي تتم علية عملية الحذف .**

**rear:- هو رير وهو العنصر الاخير الذي تم اضافته وتتم عملية الاضافه عليه.**

**العمليات التي تحدث على كيو :-**

**عملية الاضافة التي تحدث على الكيو وتسمى اينكيو****

**وعملية الحذف التي تحدث على المصفوفة وتسمى ديكيو****

**وتتكون الكيو من مصفوفه + دليلين وهما " رير + فرونت " .**

```
//abdo.solh
#include<iostream.h>
#include<conio.h>
int size=10;
 struct queue
{
int rear;
int front;
int ele[10];
}q;//********************************
void initli( queue*q)
{
q->front=0; //front =Null;
q->rear=-1;      //rear=1;

}//****************************
int insert ( queue*q,int e )
{
     if((q->rear)>size)
cout<<"the queue is full \n";
else
{q->rear=q->rear+1;
q->ele[q->rear]=e;
}
}//***************************
int delet( queue *q)
{
if(q->rear < q->front)
{cout<<"is empty \n";
 return 0;}
  else
     return q->ele[q->front++];
}//********************************
void display( queue *q)
{for(int i=q->front;i<=q->rear;i++)

cout<<q->ele[i]<<endl;
cout<<"***************";
}
//*********************************
int main()
```

```
{initli(&q);
int z;
int value;
//**********************
while(1)
{
cout<<"\n";
cout<<"enter 1 to add to queue \n";
cout<<"enter 2 to delete to queue \n";
cout<<"enter 3 to display \n";
cout<<"enter 4 to exit \n";
cin>>z;
cout<<"*********";
cout<<"\n";
//******************************
switch(z)
     {   case 1 :
      cout<<"Enter the number \n";
      cin>>value;
                 insert(&q,value);
                   break;
            case 2: cout<<"you delete value  "<< delet(&q)<<endl;
              break;
            case 3: display(&q);
             break;
            case 4: exit(0);
             break;
      }
//*********************************
}
}
//*************************************************************
```

برنامج يقوم بعملية الاسناد من داخل البرنامج من المصفوفة  والعرض  وهاذة الصورة تقوم بتوضيح عمل ال كيو:

```
#include <iostream.h>
#include <conio.h>
#define MAX 5
 class queue
{
 private:
      int t[MAX];
      int rear;
      int front;
 public:
  queue()
  {
      front=-1;
      rear=-1;
  }
  void del()
  {
     int tmp;
     if(front==-1)
```

3 1 4
Enqueue(1)
3 1 4 1
Enqueue(5)
3 1 4 1 5
Dequeue()→3
1 4 1 5
Dequeue()→1
4 1 5
Dequeue()→4
1 5

```
      {
             cout<<"Queue is Empty";
      }
      else
      {
             for(int j=0;j<=rear;j++)
             {
                    if((j+1)<=rear)
                    {
                           tmp=t[j+1];
                           t[j]=tmp;
                    }
                    else
                    {
                           rear--;

                    if(rear==-1)
                           front=-1;
                    else
                           front=0;
                    }
             }
      }
   }

void add(int item)
{
      if(front==-1 && rear==-1)
      {
             front++;
             rear++;
      }
    else
    {
             rear++;
             if(rear==MAX)
      {
                    cout<<"Queue is Full\n";
                    rear--;
                    return;
             }
      }
      t[rear]=item;

}

  void display()
  {
      if(front!=-1)
   {
      for(int i=0;i<=rear;i++)
             cout<<t[i]<<" ";
   }
   else
      cout<<"EMPTY";
  }
```

```
};

int main()
{
 queue a;
 int data[5]={32,23,45,99,24};

 cout<<"Queue before adding Elements: ";
 a.display();
 cout<<endl<<endl;

 for(int i=0;i<5;i++)
 {
   a.add(data[i]);
   cout<<"Addition Number : "<<(i+1)<<" : ";
   a.display();
   cout<<endl;
 }
 cout<<endl;
 cout<<"Queue after adding Elements: ";
 a.display();
 cout<<endl<<endl;

 for(int i=0;i<5;i++)
 {
   a.del();
   cout<<"Deletion Number : "<<(i+1)<<" : ";
   a.display();
   cout<<endl;
 }
 getch();
 return 0;
}
//************************************************************************
```

برنامج يقوم بعملية الاضافة الى الكيو ولحذف من الكيو وعرض العناصر:

```
#include <iostream.h>
//#include<stdlib.h>
#define size 10
struct queue
{
        int x[size];
        int  front;
        int  rear;
}q;
//******************************
 void initilztion(struct queue *q)
{q->front=0;
q->rear=-1;
}
//*******************************
void insert(struct queue *q,int e)
```

```
{
if((q->rear)>size)
cout<<"the queue is full \n";
else
{q->rear=q->rear+1;
q->x[q->rear]=e;}


}
//********************************
int delet(struct queue *q)
{
if(q->rear < q->front)
{cout<<"is empty \n";
 return 0;}
  else
     return q->x[q->front++];
}
//*********************************
void display(struct queue *q)
{for(int i=q->front;i<=q->rear;i++)

cout<<q->x[i]<<endl;
cout<<"****************";
}
//**********************************
int main()
{initilztion(&q);
int z;
int value;
while(1)
{
cout<<"\n";
cout<<"enter 1 to add to queue \n";
cout<<"enter 2 to delete to queue \n";
cout<<"enter 3 to display \n";
cout<<"enter 4 to exit \n";
cin>>z;
cout<<"**********";
cout<<"\n";
switch(z)
     {   case 1 :  cout<<"Enter the number \n";  cin>>value;
              insert(&q,value);   break;
```

```
            case 2: cout<<"you delete value  "<< delet(&q)<<endl;
break;
            case 3: display(&q);  break;
            case 4: exit(0); break;
      }
}
}
```

flag cirular queue add elements to flag circular queue and show them:

```
#include<iostream.h>
//#include<cstring.h>
#define max 5
struct queue
{
int data[max];
int front ;
int rear;
int count;
}qu;

void initial()
{
qu.front=qu.rear=max-1;
qu.count=0;
}
void push ()
{
if (qu.count>=max)
     cout<<"\t Circular Queue is FULL \n \n ";
     else
     {
     int info;
     cin>>info;
     qu.rear=(qu.rear+1)%max;
     qu.data[qu.rear]=info;
     qu.count++;
     }
}
int pop ()
{
if (qu.count==0)
     cout<<" Circular Queue Is Empty \n ";
     else
     {
     qu.front=(qu.front+1)%max;
     int d=qu.data[qu.front];
     qu.count--;
     cout<< d ;
```

```
      }
}
int main ()
{
initial ();
cout<<" \t\t Please Enter The Flag CQueue \n";

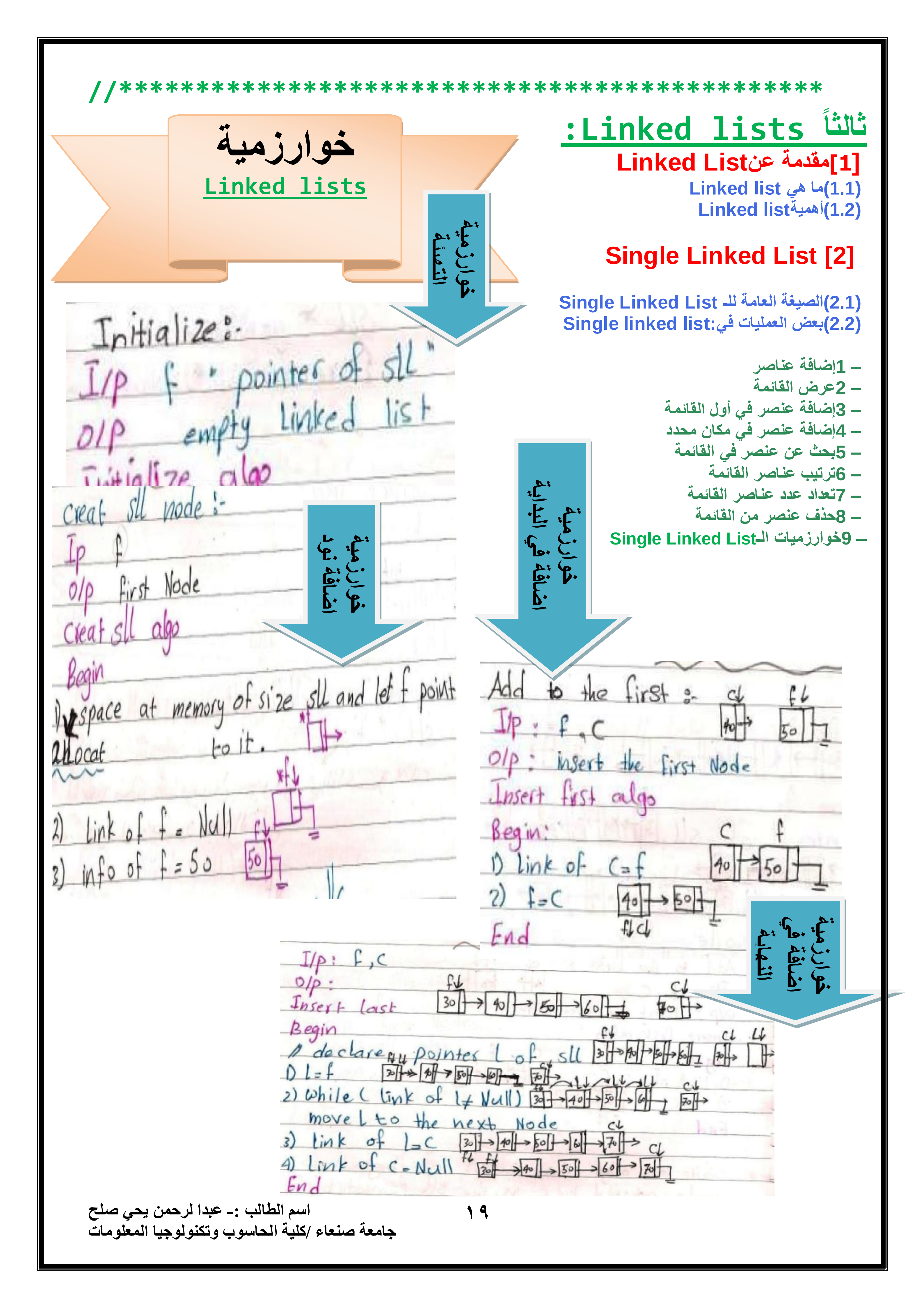
for ( int l=0;l<max;l++)
      { cout<<" \t";
      push ();
      }
cout<<" \t \t Queue after adding Elements : \n";

for (int m=0;m<max;m++)
{
cout<<"\t";
pop ();
cout<<endl;
}
}
//*********************************************************************
```

## برنامج يقوم بالاضافة والحذف والعرض و نسخ جميع محتويات المكدس الى مكدس اخر:

```
#include <iostream.h>
#include <conio.h>
int full(int count, int size)
{     return(count==size);
}
int empty(int count)
{     return count==0;
}
void add(int que[],int &tail,int &count,int size,int val)
{
      if(tail==size-1)
      tail=-1;
   //cout<<"inter value\n";
   //cin>>x;
   que[++tail]=val;
   count++;
}
int del(int que[],int &head,int &count,int size)
{
      if(head==size)
      head=0;
   count--;
   return que[head++];
}
void copy(int que1[],int que[],int size,int tail,int &tail1,int &count,int
&count1,int &head)
{     int i,t=tail,c=count;
      for(i=0;i<c;i++)
   add(que1,tail1,count1,size,del(que,head,count,size));
}
```

```
void show(int que[],int &head,int &count,int size)
{      int i,x=count;
       for(i=0;i<x;i++)
   cout<<del(que,head,count,size)<<"      ";
};
void main()
{ int c;
       int count=0,tail=-1,head=0;
       const int size=8;
       int que[size];
   int count1=0,tail1=-1,head1=0;
  //   const int size=8;
       int que1[size];
       do{ clrscr();
       cout<<"1- add\n";
       cout<<"2-del\n";
       cout<<"3-copy\n";
       cout<<"4-show\n";
       cin>>c;
       switch(c)
       {      case 1 : if(full(count,size))
                                           cout<<"full\n";
                            else
                      { int x;
                      cout<<"entr value\n";
                      cin>>x;
                                   add(que,tail,count,size,x);
                   }
                      break;
             case 2 : if(empty(count))
                                           cout<<"empty\n";
                   else
                      cout<<del(que,head,count,size)<<"\t";
                   break;
             case 3 : copy(que,que1,size,tail,tail1,count,count1,head);
                      break;
          case 4 : show(que1,head1,count1,size);
       }
       getch();
   }while(c!=0);
   getch();
}
//****************************************************************************
```



//**********************************************

## ثالثاً Linked lists:

### [1]مقدمة عنLinked List

(1.1)ما هي Linked list

(1.2)أهميةLinked list

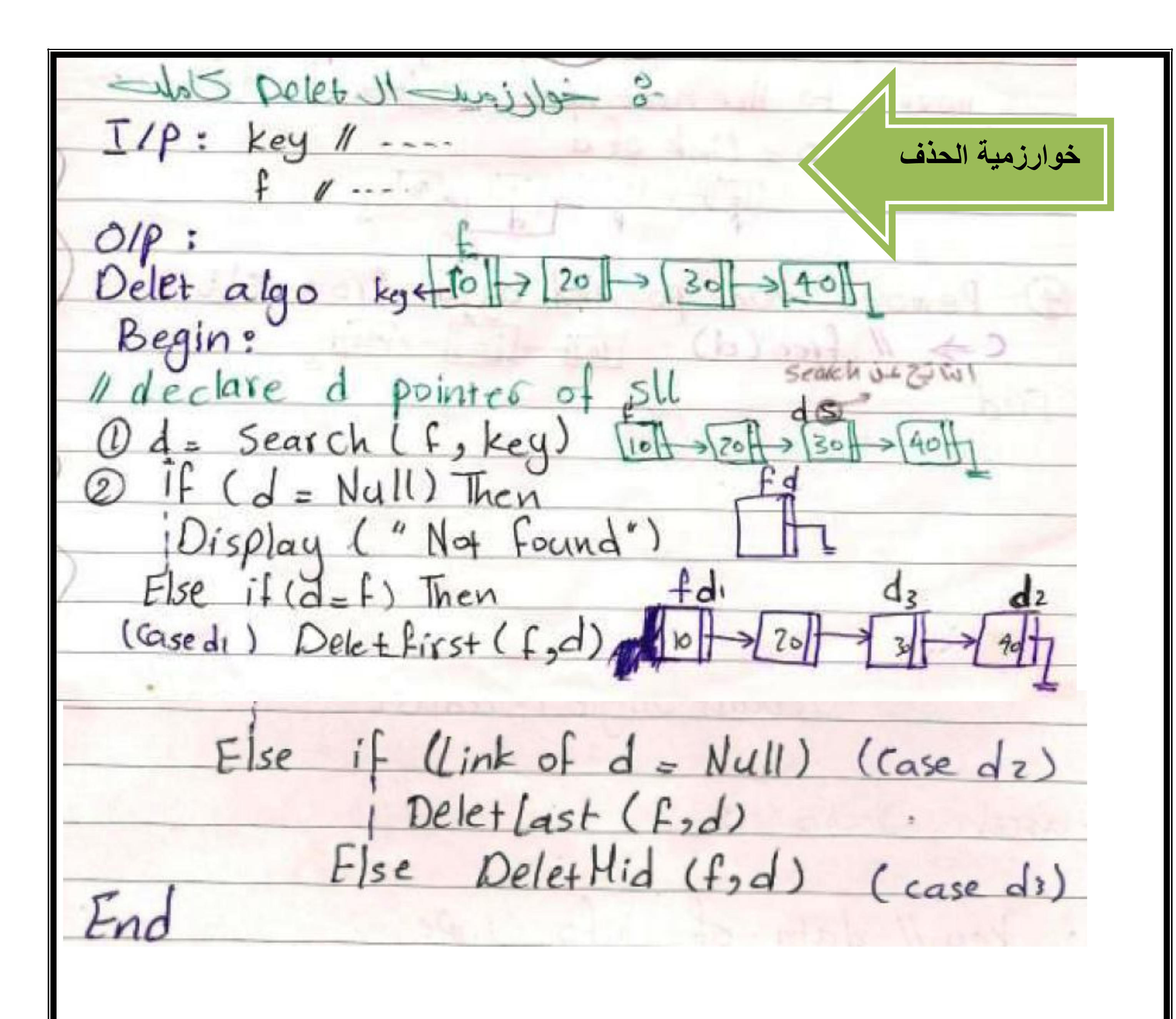
### Single Linked List [2]

(2.1)الصيغة العامة لـ Single Linked List

(2.2)بعض العمليات في:Single linked list

– 1إضافة عناصر
– 2عرض القائمة
– 3إضافة عنصر في أول القائمة
– 4إضافة عنصر في مكان محدد
– 5بحث عن عنصر في القائمة
– 6ترتيب عناصر القائمة
– 7تعداد عدد عناصر القائمة
– 8حذف عنصر من القائمة
– 9خوارزميات الـSingle Linked List

## خوارزمية Linked lists


خوارزمية الحذف

8- خوارزمية ال Delet كاملة

I/P: key // ....
f // ....

O/P:

Delet algo

Begin:

// declare d pointer of sll

① d = Search ( f, key)

② if ( d = Null) Then
Display ( " Not found")
Else if ( d = f ) Then
(case $d_1$) Delet first ( f, d)
Else if (link of d = Null) (case $d_2$)
Delet last ( f, d)
Else DeletMid ( f, d) (case $d_3$)

End



خوارزمية البحث

8- Search دالة

I/P: key // data of info type
f // ....

O/P: Memory address of node with key or Null

في حالة أن الذي تبحث عنه غير موجود

search algo

Begin

// declare object point S of sll

① S = f

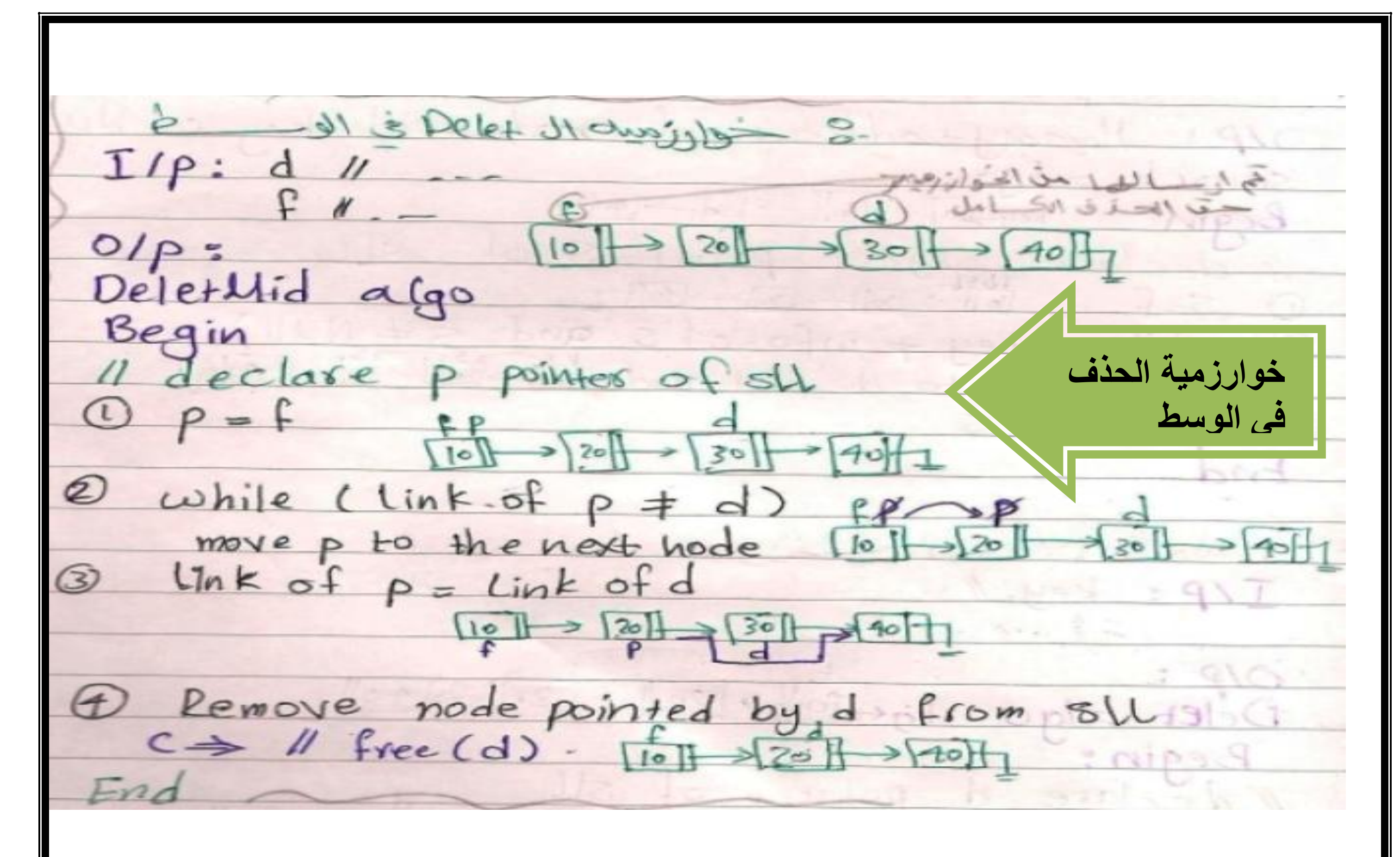
② while ( key ≠ info of s and s ≠ Null )
move s to the next node

③ Return s

End



اسم الطالب :- عبدا لرحمن يحي صلح
جامعة صنعاء /كلية الحاسوب وتكنولوجيا المعلومات




## Double Linked List [3]

### (3.1)الصيغة العامة للـDouble Linked List
### (3.2)تحويل عمليات Single linked list إلى Double Linked List

**في الحقيقة آسف للطرح موضوع على هيئة جزئيين وليس على عدة أجزاء ولذلك لأن فقرات الموضوع مترابطة جدا**
**فيما بيناهما ويصعب فهمها في أوقات متباعدة نسبيا.**
**إن شاء الله إذا فهمت هذا الموضوع سوف يسهل عليك الطريق للفهم بعض الخوارزميات الأخرى في تراكيب البيانات**
**مثل Stack و Queue وغيرها التي سوف أكتب عنها في القريب العاجل.**

## [1]مقدمة عن Linked List

### (1.1)ما هي Linked list
**هي باختصار شديد عبارة عن قائمة من البيانات مرتبطة مع بعضها البعض وغير محدودة الحجم. أي انك لا تحتاج**
**للمعرفة ما عدد البيانات المراد إدخالها إلى القائمة.**

### (1.2)أهميةLinked list
**لابد أنك استخدمت المصفوفات في العديد من برامجك وتجاربك، وكلما تعرف مصفوفة تحتاج للوضع حدود المصفوفة .**
**ولكن، ماذا لو احتجت أو احتاج مستخدم البرنامج أن يضيف عدد من البيانات يزيد عن حجم المصفوفة ؟!!.**

اظنك تعرف
ماذا سوف يحدث.
ممكن أن تحل المشكلة باستخدام المصفوفات باستعمال المؤشرات معها بنسخ المصفوفة الممتلئة إلى مصفوفة جديدة ذات حجم أكبر، و لكن مشكلة هذه الطريقة هي أنك دائما تحقق من عدد العناصر هل تجاوزات حدود المصفوفة
لكي تزيد المساحة لو احتجت لذلك.
ولكن باستخدامك **linked list** سوف يزيل عنك هم هذه المشكلة وتدخل البيانات كما يحلو لك (لكن لا تنسى حجم
الذاكرة، (لأنك لا تحتاج هل للمعرفة عدد البيانات المدخلة للقائمة فقط تقول للبيانات (حياكم الله القائمة قائمتكم) وهذا يعني بتعبير آخر عن **linked list** كما يطلق عليها البعض على أنها مصفوفة ديناميكية.

## سوف يتم شرح الاتي في ما بعد وبالتفصيل :

### ( 1) إضافة عناصر إلى القائمة-:

عند إضافة عنصر ، نتبع آلاتي:
أولا : نضع بيانات التركيب
ثانيا : إذا كان هناك عنصر آخر نشئ تركيب جديد في الذاكرة بواسطة كلمة **new** ونسند عنوان ذلك التركيب إلى
المؤشر **next** في العنصر الحالي.
ثالثا : إما إذا كان العنصر الحالي آخر عنصر في القائمة، نسند للمؤشر **next** قيمة**NULL**

### 2-عرض عناصر القائمة-:

الآن سوف نقوم للعرض عناصر القائمة ، وهي أن نقوم المرور على كل عنصر و طباعته بياناته. تتم العملية كالتالي:
أولا : نتحقق من العنصر هل هو آخر عنصر في القائمة أما لا.
ثانيا : إذا كان العنصر ليس العنصر الأخير نعرض البيانات ونتحرك إلى العنصر الذي بعده.
ثالثا : إذا كان العنصر هو العنصر الأخير نتوقف.

### 3- تعداد عدد عناصر القائمة:

مثل أي عملية تعداد أخرى ، أي نعرف عداد ومن ثم يزيد بمقدار واحد. تتم العملية التعداد كالتالي:
أولا : نتحقق من العنصر هل هو آخر عنصر في القائمة أما لا.
ثانيا : إذا كان العنصر ليس العنصر الأخير نزيد العداد بمقدار واحد.
ثالثا : إذا كان العنصر هو العنصر الأخير نتوقف.

### 4- بحث عن عنصر في القائمة:

في هذه العملية نقوم في البحث عن عنصر من عناصر القائمة و تمم العملية كالتالي:
أولا : نتحقق من العنصر هل هو آخر عنصر في القائمة أما لا.
ثانيا : إذا كان العنصر ليس العنصر الأخير نتحقق من بيانات العنصر و بيانات البحث
ثالثا : إذا كان بيانات العنصر مطابقة للبيانات البحث، نتوقف
رابع : إذا كان بيانات العنصر غير مطابقة للبيانات البحث، نتحرك للعنصر التالي.
خامسا : إذا كان العنصر هو العنصر الأخير نتوقف.

### 5- ضافة في أول القائمة-:

الآن نريد إضافة عنصر جديد في أول القائمة أي أن يكون العنصر الأول. وهذه العملية كالتالي:
أولا : نقوم بإنشاء تركيب (عنصر) جديد و إضافة بياناته.

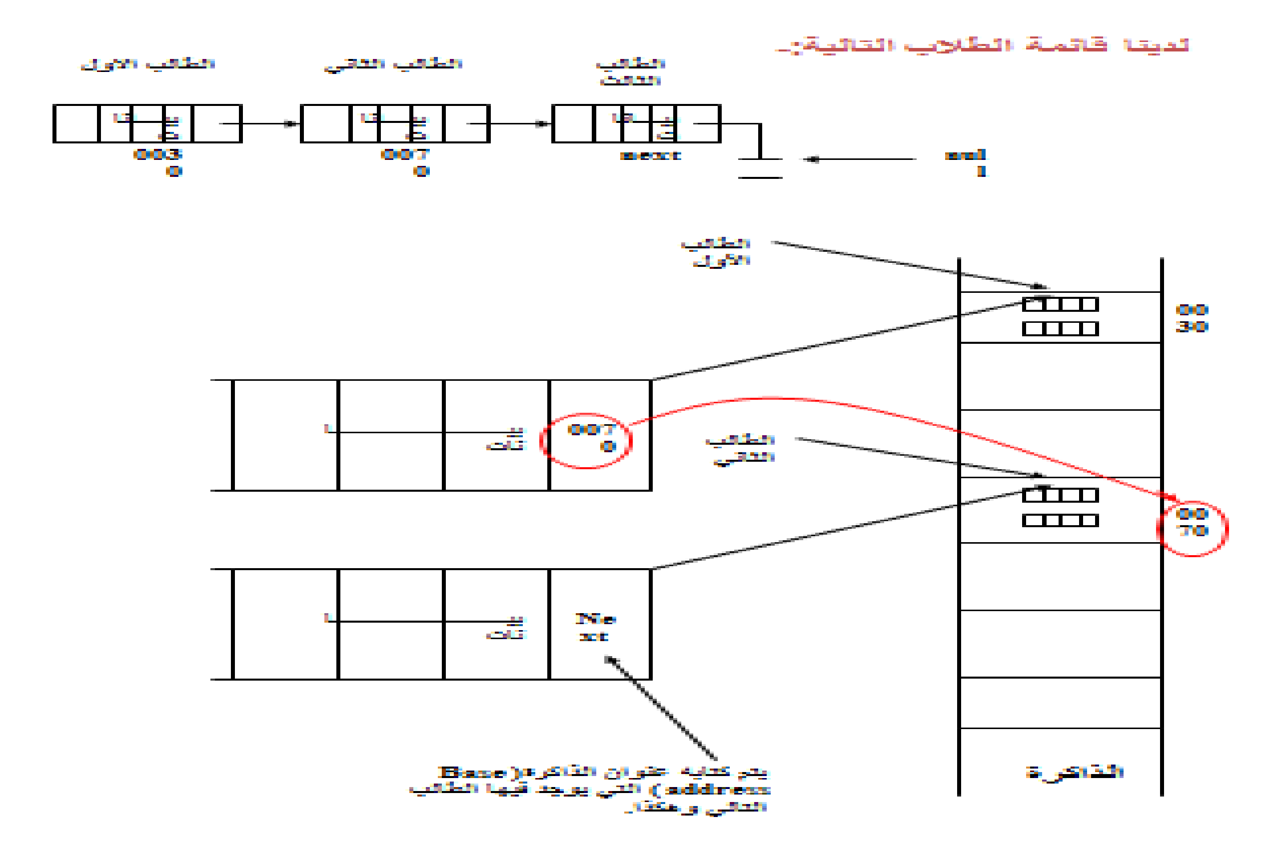
ثانيا : ثم نربط التركيب مع أول عنصر في القائمة
ثالثا : نسند للمؤشر القائمة عنوان التركيب الجديد.

# Linked list

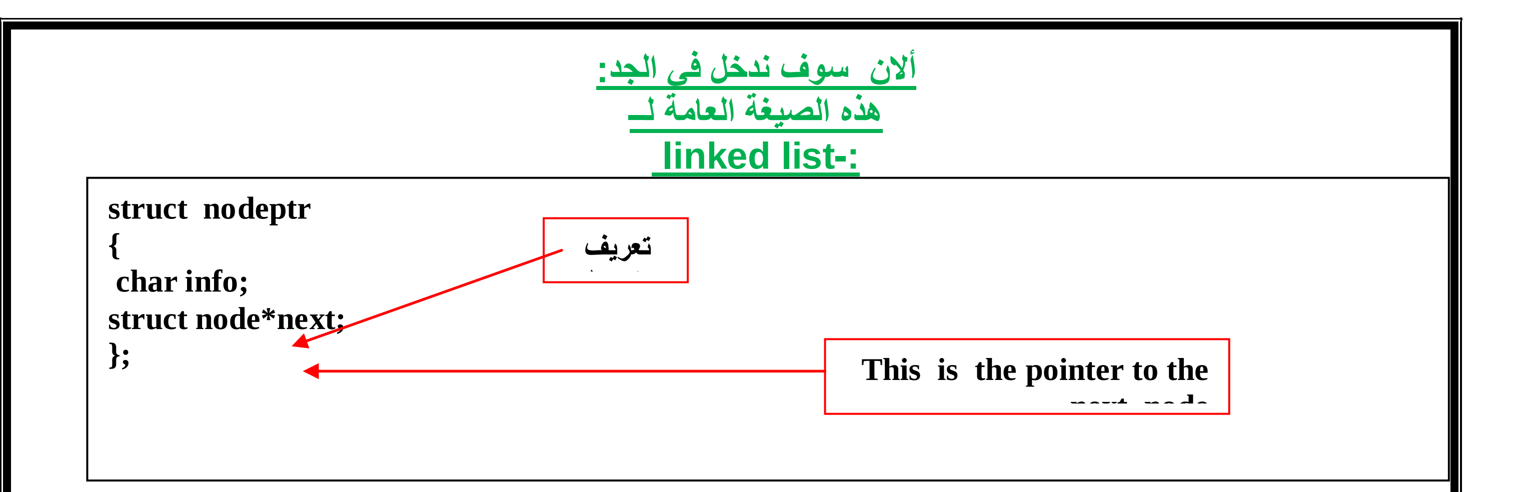
## Linked list: is a data structure wherein each element contains both a data value and a pointer to next element in the list.

### بعض مميزات الـ linked list مايلي :

تقوم بترتيب العناصر في الذاكرة حتى ولو كانت عشوائية
هنا لا نحتاج لمعرفة عدد البيانات المدخلة كما في المصفوفات التي نحدد عدد البيانات التي سوف ندخلها(نحدد حجم المصفوفة) لذالك يطلق على الـ linked list المصفوفات الديناميكية.
قد يقول البعض كيف يتم ترتيب العناصر في الذاكرة حتى ولو كانت عشوائية؟
الجواب هو أن كل عنصر يقوم بحفظ موقع العنصر الذي يليه(الذي بعدة).
وبتوضيح أكثر فلنرى مايلي:



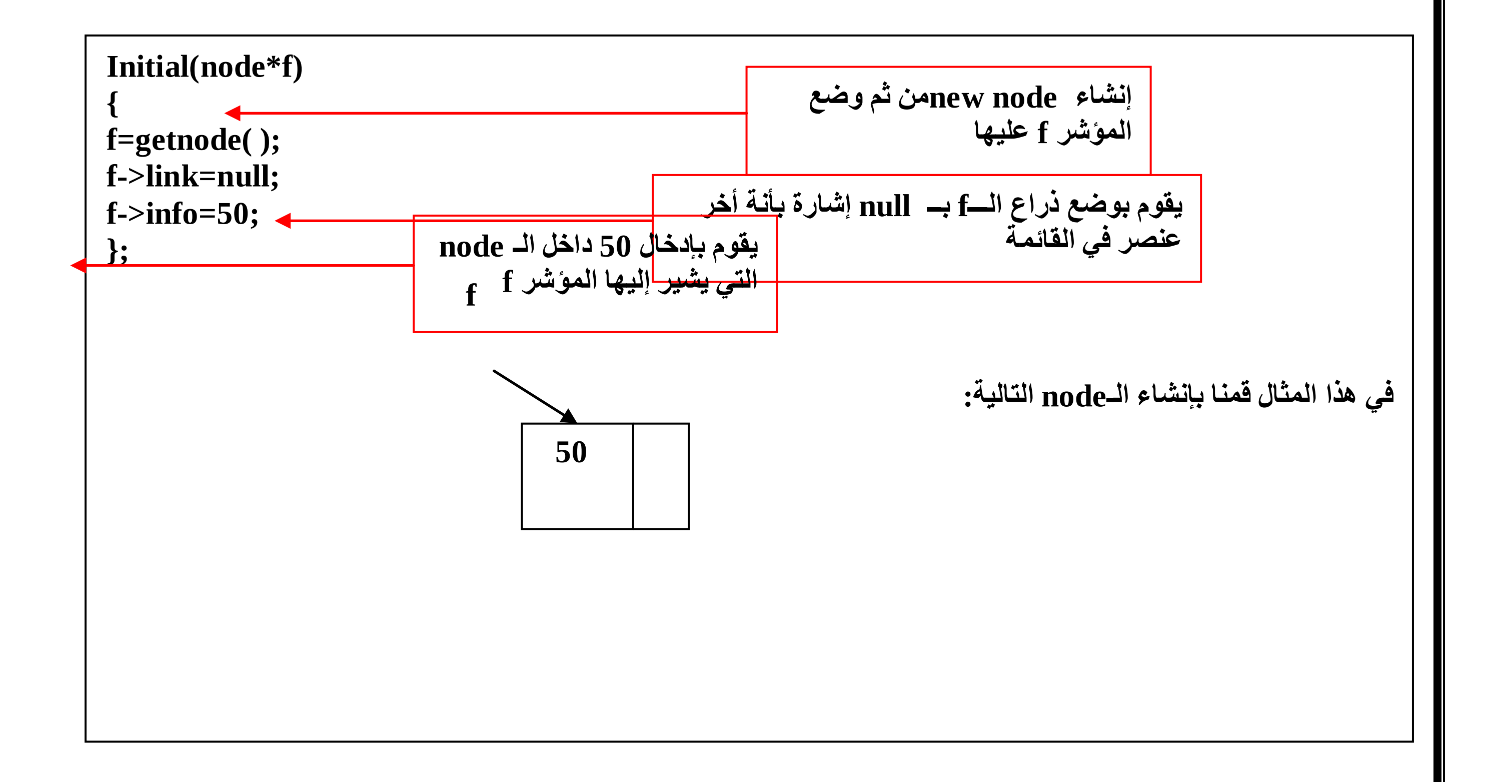
## ألان  سوف ندخل في الجد:
## هذه الصيغة العامة لـ
## linked list-:




# سوف نبدأ ألان بأهم العمليات في الــ linked list:

## أولا : عملية إنشاء node وإضافة بداخل الـــ node الرقم 50:

سيتم ذالك من خلال الخوارزمية التالية:

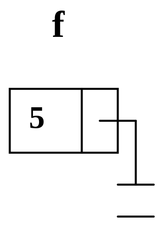





اسم الطالب :- عبدا لرحمن يحي صلح
جامعة صنعاء /كلية الحاسوب وتكنولوجيا المعلومات

## ثانياً:عملية ربط الـ node أول القائمة:

لدينا هذه الـ node ونريد ربط node جديدة ووضعها إلا ولى في القائمة



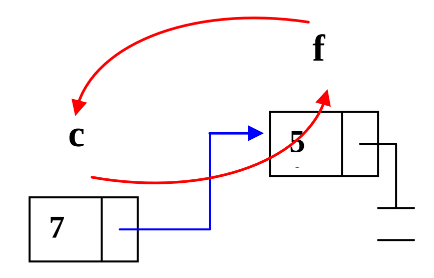

```
Insert_B(node*f)
{
node*c;
c=getnode( );
c->info=40;
c->link=f;
f=c;
}
```

إنشاء new node ثم وضع

يقوم بربط الـnode التي أنشئنها بالـ node الأولى في القائمة first

يقوم بنقل المؤشر f إلى الـ node التي ربطناها بالقائمة التي أصبحت أول node في القائمة

ستكون القائمة كما يلي:



**ثالثاً: عملية ربط node وسط القائمة (ليس بأولها وليس بآخرها):**

**هنا لدينا مثلاً بيانات احد الطلاب وكان رقمه 36 ونريد إن نضعه في القائمة حسب الرقم التسلسلي لأرقام الطلاب سيكون الحل كما يلي:**

```
Insert_l(node*f)
{
 node*c,*p;
c=getnode( );
c->info=30;
p=f;
while(c->info  > f->info  && c->info  >  p->info)
p=p->link;
c->link=p->link;
p->link=c;
}
```

وضع البيانات داخل الـnode

إنشاء new nodeو ثم وضع المؤشر c

يقوم بوضع المؤشر p في الـnode التي يشير إليها الـ f حتى يتم التنقل باستخدام المؤشر p في القائمة مما يحول دون ضياعها

يقوم بنقل المؤشر p من الـ node التي يشير إليها إلى الـ node التي تليها

يقوم بتنفيذها بعد توقف wile طبعاً و هذا السطر يقوم بالتالي:
يقوم بربط ذراع الـnode المراد ربطها بالـ node التي يشير إليها ذراع الـ p

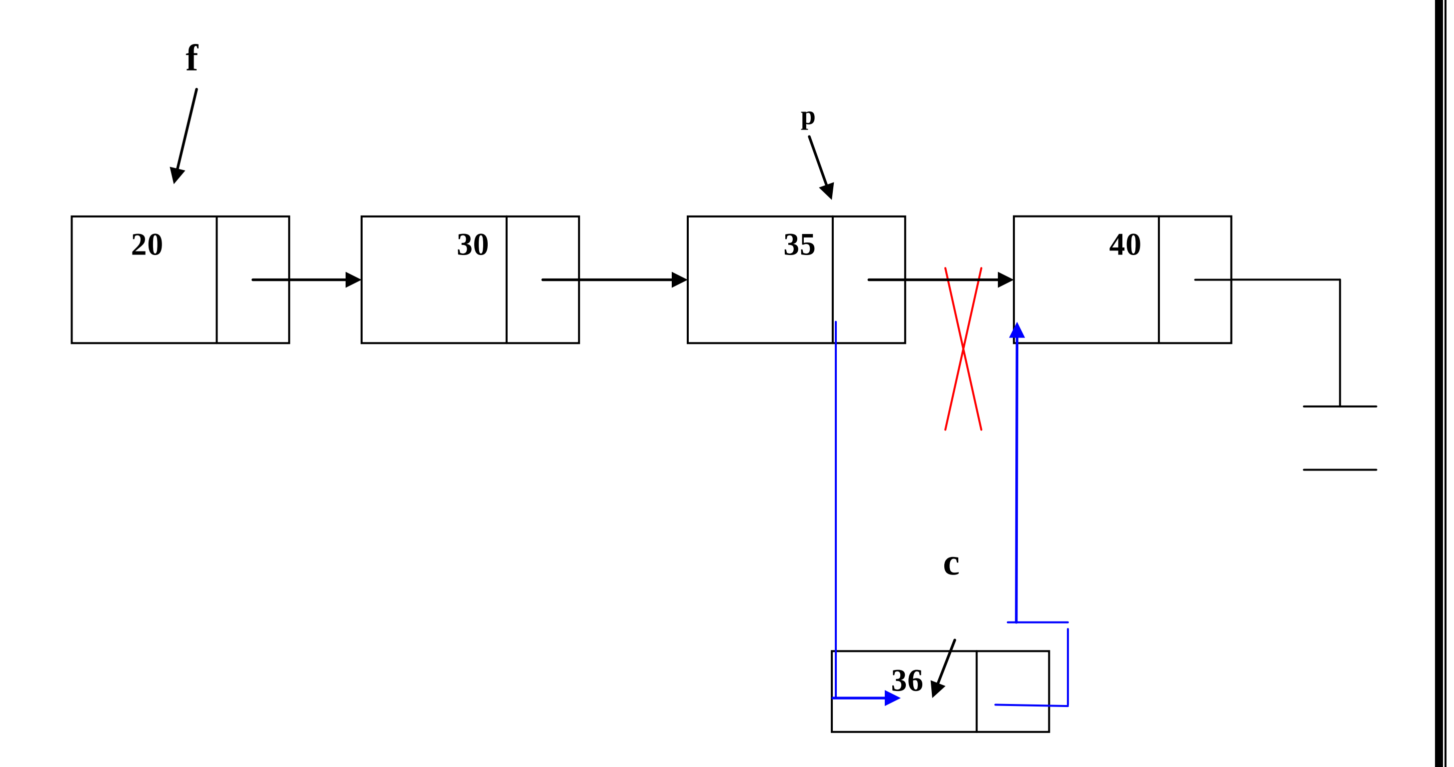
يقوم بنقل ذراع المؤشر p من الـnode التي كان يشير إليها إلى الـ nodeالتي تم ربطها(الـc)

هذا السطر يقوم بالتالي: طالما (c->info >f->info) معلومات الـnode التي يشير إليها المؤشر c اكبر من معلومات الـnode التي يشير إليها المؤشر)&&( f و ((c->info>p->linked->info معلومات الـnode التي يشير إليها المؤشر c (معلومات الـ node المراد ربطها بالقائمة) اكبر من معلومات الـ node التي يشير إليها ذراع المؤشر p



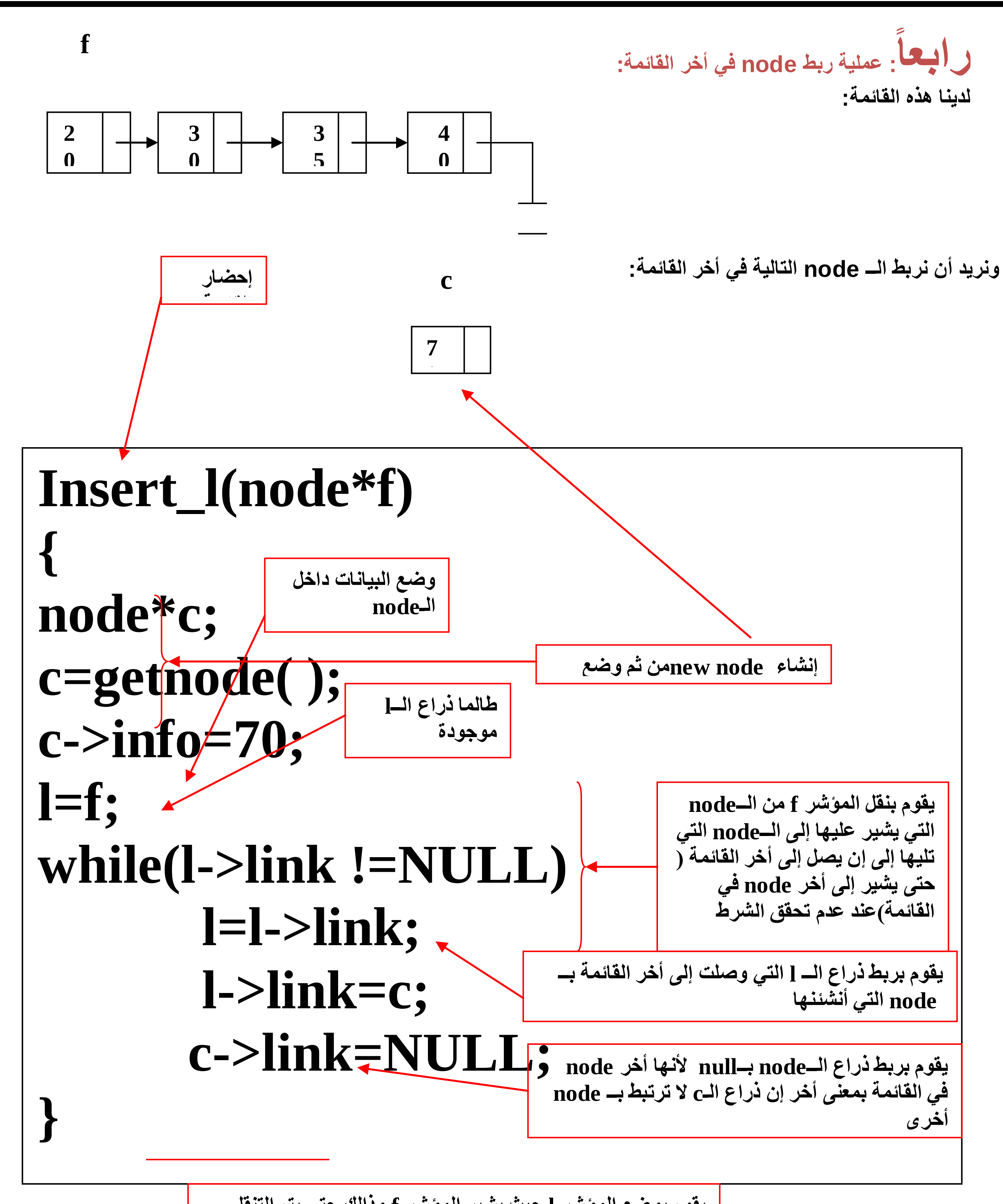
اسم الطالب :- عبدا لرحمن يحي صلح
جامعة صنعاء /كلية الحاسوب وتكنولوجيا المعلومات

## بعد هذا كله راح تكون القائمة بهذا الشكل:



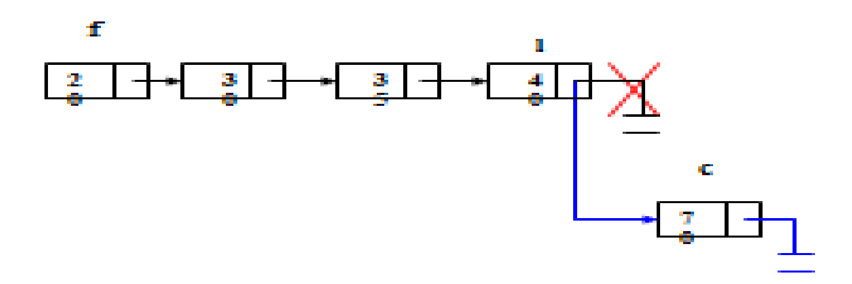
## رابعاً: عملية ربط node في أخر القائمة:

لدينا هذه القائمة:




يقوم بوضع المؤشر l حيث يشير المؤشر f وذالك حتى يتم التنقل باستخدام المؤشر l إلى أخر القائمة وذالك تفادين لعدم ضياع القائمة حيث أذا تم نقل العنصر f من مكانة في أول عنصر في القائمة فإننا سوف نفقد القائمة بأكلمها وذالك خطأ فادح جداً.

بعد هذا كله راح تكون القائمة
بهذا الشكل:




## خامساً:عملية زيارة القائمة

ألان سوف نقوم بعملية المرور على كل عناصر القائمة وطبع ما يوجد بداخلها من معلومات:

```
Traverse(node*f)
{
node*c=f;
while(c->link  !=NULL)
{
cout<<c->info;
c=c->link;
}
}
```

طالما ذراع المؤشر c لا تساوي null

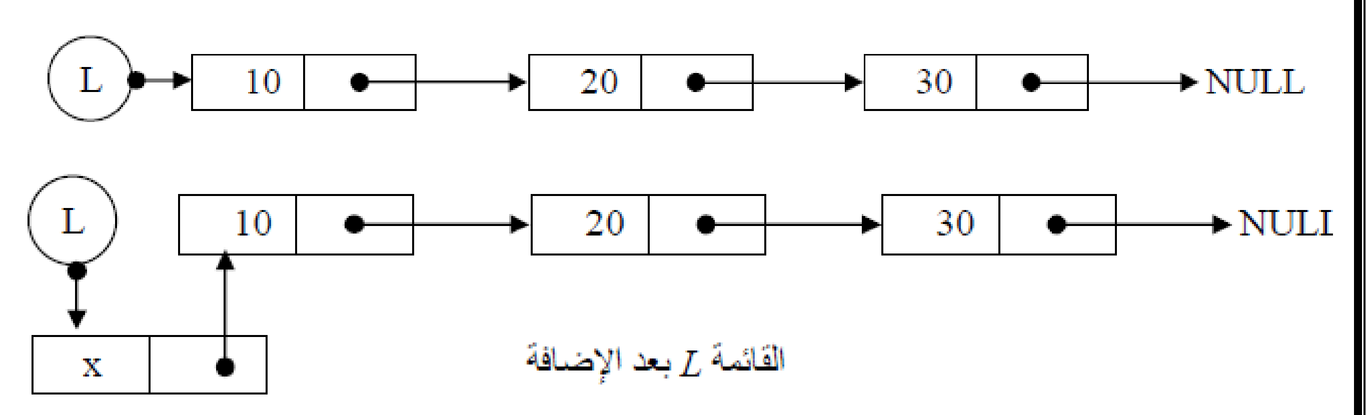
يقوم بطبع المعلومات الموجودة في الـnode التي تشير إليها الـc

يقوم بنقل ذراع المؤشر c الى الـ node التي تليه

وضعنا القوسين ليتم تنفيذ ما بينهما بالكامل اذا تحقق شرط while

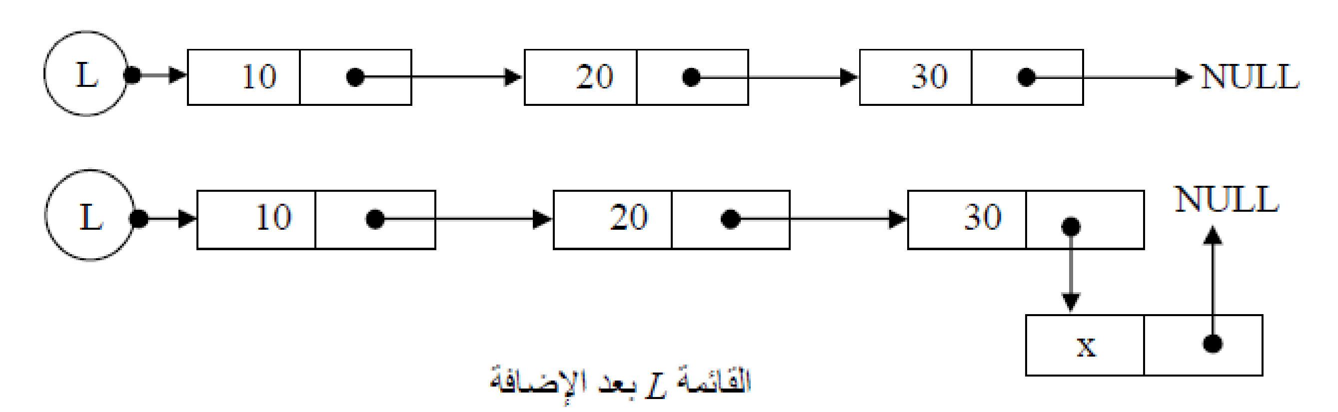
*****************************************************************************************

**دالة تقوم بإضافة عنصر في بدية القائمة المتصلة:**




```
Node* AddItemFront(int x, Node *L)
{
Node *P;
P = new(Node);
P -> Data = x;
P -> Next = L;
L = P;
return L;
}
```

## دالة تقوم بإضافة عنصر في نهاية القائمة المتصلة:



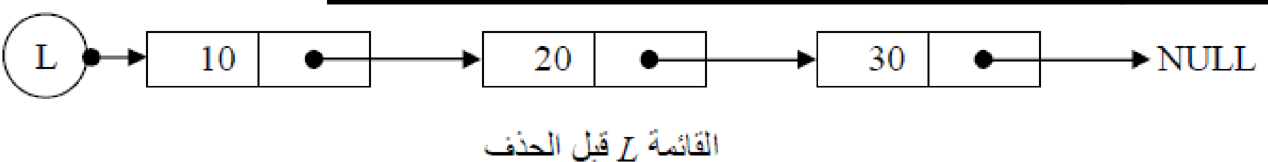

القائمة *L* بعد الإضافة

```
Node* AddItemRear(int x, Node *L)
{
Node *P, *Q;
if (L = = NULL)
{
Q = new(Node);
Q -> Data = x;
Q -> Next = NULL;
L = Q;
}
else
{
P = L;
while (P->Next != NULL)
{
P = P -> Next;
}
Q = new(Node);
Q -> Data = x;
Q -> Next = NULL;
P -> Next = Q;
}
return L;
}
```

## دالة تقوم بالبحث عن أي عنصر داخل القائمة المتصلة :

```
bool SearchItem(int x, Node *L)
{
Node *P = L;
if (P == NULL)
return false;
else
if (P -> Data == x)
     return true;
else
return SearchItem(x, P -> Next);
}
```

## دالة تقوم بالحذف أي عنصر داخل القائمة المتصلة:



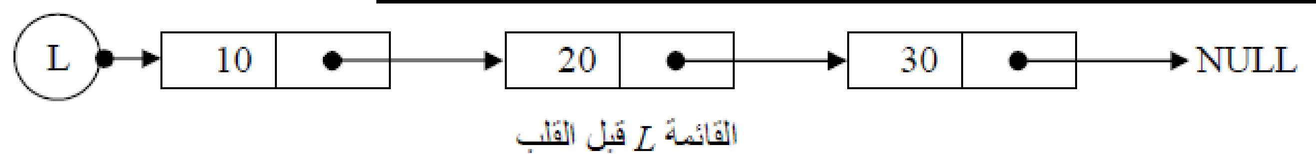

القائمة *L* قبل الحذف



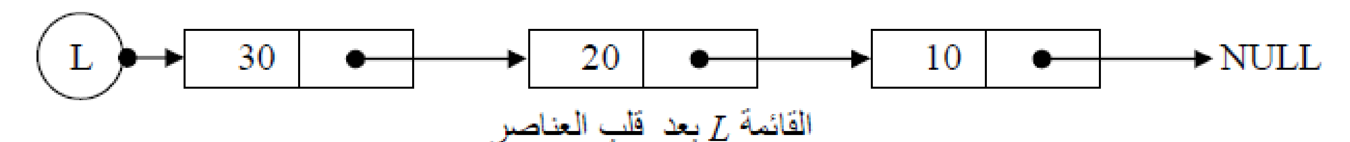

القائمة *L* بعد حذف العنصر 20

```
Node* DeleteItem(int x, Node *L)
{
Node *P = L, *Q;
if (P -> Data == x)
{
L = L ->Next;
delete P;
}
else
{
while ((P->Next)->Data != x)
P = P -> Next;
Q = P -> Next;
P -> Next = Q -> Next;
delete Q;
}
return L;
}
```

## دالة تقوم بقلب عناصر القائمة المتصلة:



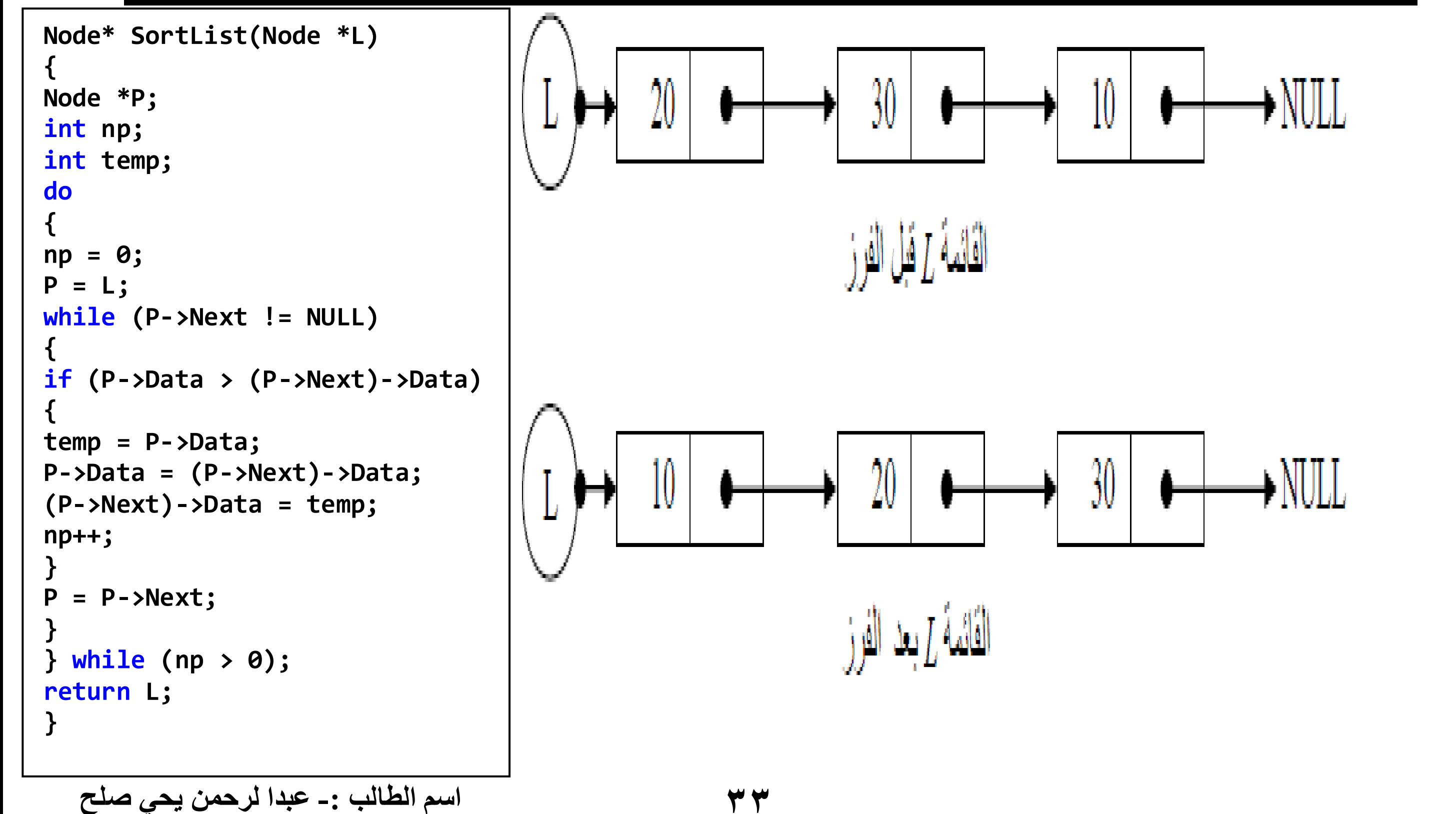

القائمة L قبل القلب




القائمة L بعد قلب العناصر

```
Node* InvertList(Node *L)
{
Node *P = L;
Node *Q = NULL;
Node *R;
while (P != NULL)
{
R = new(Node);
R -> Data = P->Data ;
R -> Next = Q;
Q = R;
P = P->Next;
}
return Q;
}
```

## دالة تقوم بفرز القائمة المتصلة باستخدام طريقة الفرز الفقاعي:



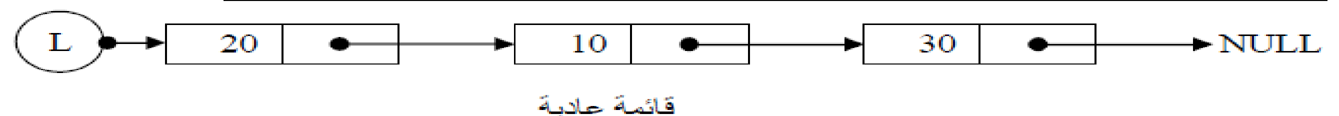
```
Node* SortList(Node *L)
{
Node *P;
int np;
int temp;
do
{
np = 0;
P = L;
while (P->Next != NULL)
{
if (P->Data > (P->Next)->Data)
{
temp = P->Data;
P->Data = (P->Next)->Data;
(P->Next)->Data = temp;
np++;
}
P = P->Next;
}
} while (np > 0);
return L;
}
```



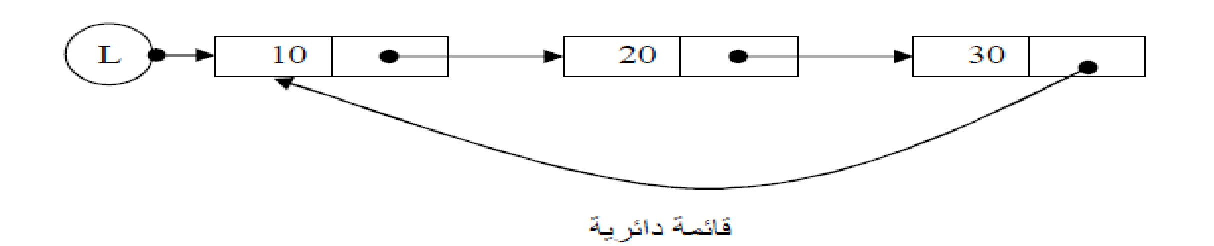
القائمة L قبل الفرز

القائمة L بعد الفرز

# دالة تقوم بتحويل القائمة العادية الى قائمة دائرية:




قائمة عادية




قائمة دائرية

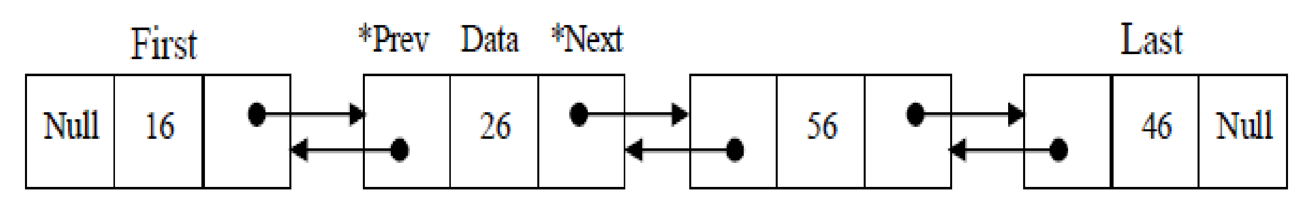
```
Node* Circular(Node *L)
{
Node *P;
if (L != NULL)
{
P = L;
while (P->Next != NULL)
P = P->Next;
P -> Next = L;
}
return L;
}
```

# دالة تقوم بطباعة جميع عناصر القائمة الدائرية:

```
void PrintCircular(Node *Q)
{
Node *P;
if (Q != NULL)
{
cout<<Q->Data<<"\t";
P = Q->Next;
while (P != Q)
{
cout<<P->Data<<"\t";
P = P->Next;
}
}
}
```

## القوائم المتصلة ذات الاتجاهين(Doubly-linked lists):

## الشكل العام (Doubly-linked lists):




قائمة ذات اتجاهين

عند تعريف القائمة لابد من اعلان عن مؤشر الى اول اقائمة ومؤشر نحو اخر القائمة مثل:

```
struct Node
{
Node *Prev;
int D ata;
Node *N ext;
}
struct Dbl_list
{
Node *F irst;
Node *L ast;
}
```

//*****************************

**برنامج يقوم با ضافة عنصر في بداية القائمة واضافة عنصر في نهاية القائمة وعرض عناصر القائمة من الاول الى الاخير وعرض عناصر القائمة من الاخير الى الاول وحذف أي عنصر من القائمة:**

```
#include <iostream>
using namespace std;
struct Node
{
```

```
Node *Prev;
int Data;
Node *Next;
};
struct Dbl_List
{
Node *First;
Node *Last;
};
Dbl_List AddFront(int x, Dbl_List L)
{
Node *P ;
if (L.First == NULL)
{
P = new(Node);
P->Prev = NULL;
P->Data = x;
P->Next = NULL;
L.First = P;
L.Last = P;
}
else
{
P = new(Node);
P->Prev = NULL;
P->Data = x;
P->Next = L.First;
L.First->Prev = P;
L.First = P;
}
return L;
```

```
}
Dbl_List AddRear(int x, Dbl_List L)
{
Node *P ;
if (L.First == NULL)
{
P = new(Node);
P->Prev = NULL;
P->Data = x;
P->Next = NULL;
L.First = P;
L.Last = P;
}
else
{
P = new(Node);
P->Prev = L.Last;
P->Data = x;
P->Next = NULL;
L.Last->Next = P;
L.Last = P;
}
return L;
}
void ShowListFarward(Dbl_List L)
{
Node * P = L.First;
while (P != NULL)
{
cout<<P->Data<<"\t";
P = P->Next;
```

```
}
cout<<endl;
}
void ShowListBackward(Dbl_List L)
{
Node * P = L.Last;
while (P != NULL)
{
cout<<P->Data<<"\t";
P = P->Prev;
}
cout<<endl;
}
Dbl_List DeleteItem(int x, Dbl_List L)
{
Node *P ;
if (L.First->Data == x)
{
P = L.First;
L.First = L.First ->Next ;
L.First ->Prev = NULL;
delete P;
}
else
{
if (L.Last->Data == x)
{
   P = L.Last;
L.Last = L.Last ->Prev ;
L.Last ->Next = NULL;
delete P;
```

```
}
else
{
P = L.First;
while((P != NULL)&&(P->Data != x))
P = P -> Next;
if (P == NULL)
cout<<"Not Found.."<<endl;
else
{
(P->Prev)->Next = P->Next;
(P->Next)->Prev = P->Prev;
delete P;
}
}
}
return L;
}
 main()
{
Dbl_List L;
L.First = NULL;
L.Last = NULL;
L = AddFront(26,L);
L = AddFront(16,L);
L = AddRear(36,L);
L = DeleteItem(26,L);
ShowListFarward(L);
}
```

**هاذا برنامج الستاك والكيو والنكد لست ولكن با اكواد مختصرة لم نكتب جميع الاكواد ولكن كتبناء القليل منهاء:**

```
#include<iostream.h>
//*******************************
const int size=5;
//*******************************
struct stack
{int top;
int item[size];
int rear;
int front;
stack *First;
stack *Last;
stack *Prev;
int Data;
stack *Next;
}ps;
//*****************************
/*struct Dbl_List
{
Node *First;
Node *Last;
};
//*****************************
struct Node
{Node *Prev;
int Data;
Node *Next;
};*/
//*****************************
void initial(struct stack *ps)
{
ps->top=-1;
ps->front=0; //front =Null;
ps->rear=-1;
}
//******************************
void push(struct stack *ps,int m)
{if(ps->top<(size-1))
{(ps->top)++;
ps->item[ps->top]=m;}
else
cout<<"full\n";}
```

```
//*********************************
int insert ( stack*ps,int e )
{
if((ps->rear)>size)
cout<<"the queue is full \n";
else
{ps->rear=ps->rear+1;
ps->item[ps->rear]=e;
}
}//***************************
void pop(struct stack *ps)
{if(ps->top>(-1))
{cout<<ps->item[ps->top]<<endl;
(ps->top)--;}
else
cout<<"empty\n";
}
//*********************************
int delet(  stack *ps)
{
if(ps->rear < ps->front)
{cout<<"is empty \n";
return 0;}
else
return ps->item[ps->front++];
}//*********************************
void display( stack *ps)
{for(int i=ps->front;i<=ps->rear;i++)
cout<<ps->item[i]<<endl;
cout<<"\n***************\n";
}
//*********************************
void sho( stack *ps)
{
for(int i= 0;i<ps->top ;i++)
{cout<<ps->item[i];}
cout<<"empty\n";
}//*********************************
stack AddFront(int x, stack L)
{
stack *P ;
if (L.First == NULL)
{
P = new(stack);
```

```
P->Prev = NULL;
P->Data = x;
P->Next = NULL;
L.First = P;
L.Last = P;
}
else
{
P = new(stack);
P->Prev = NULL;
P->Data = x;
P->Next = L.First;
L.First->Prev = P;
L.First = P;
}
return L;
}
//***********************************
stack AddRear(int x, stack L)
{
stack *P ;
if (L.First == NULL)
{
P = new(stack);
P->Prev = NULL;
P->Data = x;
P->Next = NULL;
L.First = P;
L.Last = P;
}
else
{
P = new(stack);
P->Prev = L.Last;
P->Data = x;
P->Next = NULL;
L.Last->Next = P;
L.Last = P;
}
return L;
}
//***********************************
void ShowListFarward(stack L)
{
```

```
stack * P = L.First;
while (P != NULL)
{
cout<<P->Data<<"\t";
P = P->Next;
}
cout<<endl;
}
//**********************************
void ShowListBackward(stack L)
{
stack * P = L.Last;
while (P != NULL)
{
cout<<P->Data<<"\t";
P = P->Prev;
}
cout<<endl;
}
//***********************************
stack DeleteItem(int x, stack L)
{
stack *P ;
if (L.First->Data == x)
{
P = L.First;
L.First = L.First ->Next ;
L.First ->Prev = NULL;
delete P;
}
else
{
if (L.Last->Data == x)
{
P = L.Last;
L.Last = L.Last ->Prev ;
L.Last ->Next = NULL;
delete P;
}
else
{
P = L.First;
while((P != NULL)&&(P->Data != x))
P = P -> Next;
```

```
if (P == NULL)
cout<<"Not Found.."<<endl;
else
{
(P->Prev)->Next = P->Next;
(P->Next)->Prev = P->Prev;
delete P;
}
}
}
return L;
}
/***********************************
int SearchItem(int x, stack *L)
{cin>>x;
stack *P = L;
if (P == NULL)
return false;
else
if (P -> Data == x)
return true;
else
return SearchItem(x, P -> Next);
}
*/
//***********************************
//دالة تقوم بقلب العناصر المدخلة مشابهة لعمل البوب
stack* InvertList(stack *L)
{
stack *P = L;
stack *Q = NULL;
stack *R;
while (P != NULL)
{
R = new(stack);
R -> Data = P->Data ;
R -> Next = Q;
Q = R;
P = P->Next;
}
return Q;
}
//***********************************
int main()
```

```
{
     stack L;
L.First = NULL;
L.Last = NULL;
int n, op, x;;
initial(&ps);
do{
cout<<"\stack \n\n1 to push and \n2 to pop\n6show
stacke\n\nqueue\n\n3 insert queue\n4 delete queue \n5 display
queue\n \nSingl linked list\n\n7 add to front\n8 add to front\n9
ShowListFarward\n10 DeleteItem of linked list\n";
cin>>op;
switch(op)
{case 1:;
{cin>>n;
push(&ps,n);
}
break;
case 2:
pop(&ps);
break;
case 3:
cout<<"Enter the number \n";
cin>>n;
insert(&ps,n);
break;
case 4:
cout<<"you delete value "<< delet(&ps)<<endl;
break;
case 5:
 display(&ps);
break;
case 6:
sho(&ps);break;
case 7:
cout<<"pleas  add 5 to front\n";
for(int i=0;i<=size;i++)
{x=i;
cin>>x;}
//L.First = NULL;
//L.Last = NULL;}
AddFront(x,L);
break;
case 8:
```

```
cout<<"pleas  add 5 to Rear\n";
for(int i=0;i<=size;i++)
{x=i;
cin>>x;}
AddRear( x,L);
break;
case 9:
ShowListFarward(L);
break;
case 10:
cout<<"Enter the number delete  of singel linked list\n";
cin>>x;
DeleteItem(*&x,L);
break;
/*case 11: //cin>>x;
I//nvertList(stack);
break;*/
default:
cout<<"error";
}
}
while(op!=0);
return 0; }
//************************************
```

**برنامج يقوم بالإضافة في البداية والنهاية وقبل أي رقم ويقوم بعرض العمليات المدخلة والاضافة بعد أي رقم  والحذف من البداية والنهاية ومن الوسط والحذف من أي مكان  وعرض العناصر المتصلة بعد عملية الحذف :**

```
#include<iostream.h>
#include<conio.h>
#include<alloc.h>
struct dnode
{
int data;
struct dnode*llink,*rlink;
};
struct dnode*f;
 dnode *intl()
{
f=NULL;
return f;
```

```
}
dnode*creatnode()
{
return((dnode*)malloc(sizeof(dnode)));
}
//***********************************
void addbeg(dnode*&f,dnode*n)
{
if(f==NULL)
{
n->rlink=f;
n->llink=NULL;
f=n;
}
else
{
n->rlink=f;
n->llink=NULL;
f->llink=n;
f=n;
}
}
void addend(dnode*f,dnode*n)
{
dnode*p=NULL;
p=f;
while(p->rlink!=NULL)
p=p->rlink;
n->llink=p;
n->rlink=NULL;
p->rlink=n;
}
void addaft(dnode*f,dnode*n,int e)
{
dnode*p=NULL;
p=f;
while(p!=NULL&& p->data!=e)
p=p->rlink;
if(p==NULL)
cout<<" \n data you want to add after it not found \n";
else
{
n->llink=p;
n->rlink=p->rlink;
```

```
p->rlink->llink=n;
p->rlink=n;
}
}
void visilist(dnode*f)
{
dnode*p=NULL;
p=f;
if(p==NULL)
cout<<" \n there is no node in the list \n";
{
while(p!=NULL)
{
cout<<p->data;
p=p->rlink;
}
}
}
void revlist(dnode*f)
{
dnode*p=NULL;
p=f;
if(p==NULL)
cout<<" \n there is no nodes in the list \n";
else
while(p->rlink!=NULL)
p=p->rlink;
while(p!=NULL)
{
cout<<p->data;
p=p->llink;
}
}
//************************************
void addbef(dnode*f,dnode*n,int e)
{
dnode*p=NULL;
p=f;
while( p!=NULL&&p->data!=e)
p=p->rlink;
if(p==NULL)
cout<<"\n data you want add  befor it not found \n";
else
{
```

```
n->llink=p->llink;
n->rlink=p;
p->llink->rlink=n;
p->llink=n;
}
}
void dbeg(dnode*&f)
{
dnode*d=NULL;
d=f;
f=f->rlink;
f->llink=NULL;
free(d);
}
void dend(dnode*f)
{
dnode*p=NULL;
p=f;
while(p->rlink!=NULL)
p=p->rlink;
p->llink->rlink=NULL;
free(p);
}
void dmid(dnode*f,int e)
{
dnode*d=NULL;
d=f;
while(d->data!=e&& d->rlink!=NULL)
d=d->rlink;
if( d->rlink==NULL)
cout<<" \n data that you want to delete it not found  in midlist
\n";
else
{
d->llink->rlink=d->rlink;
d->rlink->llink=d->llink;
free(d);
}
}
void danywhere(dnode*&f,int key)
{
dnode*d=NULL;
dnode*pf=NULL;
dnode*p=NULL;
```

```
p=f;
d=f;
pf=f;
int found=0;
while(d!=NULL&&!found)
{
if(d->data==key)
found=1;
else
{
p=f;
d=d->rlink;
}
}
if(!found&&d==NULL)
cout<<"\n node that you want to delete it not found \n";
if(found)
{
if(pf==d)
{
f=f->rlink;
f->llink=NULL;
free(d);
}
else
{
d->llink->rlink=d->rlink;
d->rlink->llink=d->llink;
free(d);
}
}
}
main()
{
//clrscr();
dnode*f=intl();
dnode*n=NULL;
int x;

int op;
do
{
cout<<"\n enter 1 to addbegin and \n 2 to addend  \n 3 to addaft
\n 4  to display \n 5 to exit \n 6 to add befor specific data \n
```

```
7 to delete from begin \n 8 to delete from end\n 9 to delete from
mid list \n 10 to delete any where \n 11 to show list from last
\n";
cin>>op;
switch(op)
{
case 1:
n=creatnode();
cout<<"enter information";
cin>>n->data;
addbeg(f,n);
break;
case 2:
n=creatnode();
cout<<"enter information";
cin>>n->data;
addend(f,n);
break;
case 3:
n=creatnode();
cout<<"enter data";
cin>>n->data;
cout<<"enter data that you want to add after it";
cin>>x;
addaft(f,n,x);
break;
case 4:
visilist(f);
break;
case 6:
int y;
n=creatnode();
cout<<"enter information of node \n";
cin>>n->data;
cout<<"enter data you want add befor it \n";
cin>>y;
addbef(f,n,y);
break;
case 7:
dbeg(f);
break;
case 8:
dend(f);
break;
```

```
case 9:
int z;
cout<<" \n enter data you wnat to delete from list \n ";
cin>>z;
dmid(f,z);
break;
case 10:
int t;
cout<<"\n enter data you want delete it \n";
cin>>t;
danywhere(f,t);
break;
case 11:
revlist(f);
break;
}
}while(op!=5);
}
```

//*************************************

## برنامج يقوم بعملية الاضافة في البداية والنهاية وقبل أي رقم ويقوم بالعرض ويقوم بعملية الاضافة بعد أي رقم والحذف من البداية والنهاية والوسط وترتيب العناصر وتقسيم القائمة وايجاد مجموع الاعداد الزوجية والفردية:

```
#include<iostream.h>
//using namespace std;
#include<conio.h>
#include<alloc.h>
struct sll
{
int data;
struct sll*llink;
};
sll*f;
sll*creatnode()
{
return((sll*)malloc(sizeof(sll)));
}
sll*intl()
{
f=NULL;
return f;
}
void addbeg(sll*&f,sll*n)
{
```

```
n->llink=f;
f=n;
}
void addend(sll*f,sll*n)
{
sll*p=NULL;
p=f;
while(p->llink!=NULL)
p=p->llink;
p->llink=n;
n->llink=NULL;
}
void addaft(sll*f,sll*n,int e)
{
sll*p=NULL;
p=f;
while(p->data!=e&&p!=NULL)
p=p->llink;
if(p!=NULL)
{
n->llink=p->llink;
p->llink=n;
}
else
cout<<"the node you want to add after it not found \n";
}
void visitlist(sll*f)
{
sll*p=NULL;
p=f;
while(p!=NULL)
{
cout<<p->data;
p=p->llink;
}
}
void addbef(sll*f,sll*n,int e)
{
sll*p=NULL;
sll*old=NULL;
p=f;
while(p->data!=e&&p!=NULL)
{
old=p;
p=p->llink;
}
if(p!=NULL)
{
n->llink=p;
old->llink=n;
}
else
cout<<"the node you want to add befor it not found \n";
}

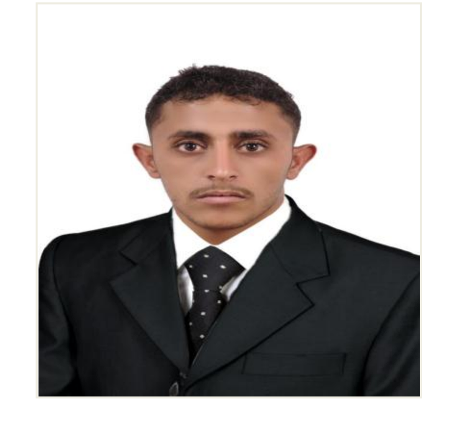
void dbeg(sll*&f)
{
```

```
sll*d=NULL;
d=f;
f=f->llink;
free(d);
}
void dend(sll*f)
{
sll*p=NULL;
sll*d=NULL;
p=f;
while(p->llink!=NULL)
{
d=p;
p=p->llink;
}
d->llink=NULL;
}
void dlend(sll*f)
{
      sll*d;
      sll*p;
      d=f;
      p=f;
      while(p->llink!=NULL)
      {
            d=p;
            p=p->llink;
      }
      d->llink=NULL;
      free(p);
}
void dmid(sll*f,int key)
{
sll*p=NULL;
sll*d=NULL;
p=f;
while(p->llink->data!=key&&p!=NULL)
p=p->llink;
d=p->llink;
if(p!=NULL)
p->llink=d->llink;
else
cout<<"\n data you want to delete it not found \n";
}
//********************
void odev(sll*&f,sll*n)
{
sll*p=f;
if(n->data%2==0)
{
n->llink=f;
f=n;
}
else
{
if(f==NULL)
{
```

```
n->llink=f;
f=n;
}
else
{
p=f;
while(p->llink!=NULL)
p=p->llink;
n->llink=NULL;
p->llink=n;
p=n;
}
}
}
//*********************************
void div_list(sll*f,int e)
{
sll*p=NULL;
sll*p2=NULL;
sll*f2=NULL;
p=f;
while(p->data!=e)
p=p->llink;
f2=p->llink;
p2=p->llink;
p->llink=NULL;
cout<<"\n first list \n";
p=f;
while(p!=NULL)
{
cout<<p->data;
p=p->llink;
}
cout<<"\n second list is \n";
p2=f2;
while(p2!=NULL)
{
cout<<p2->data;
p2=p2->llink;
}
}
//*************************
void findsum(sll*f)
{
sll*p=NULL;
p=f;
int sumeven=0;
int sumodd=0;
while(p!=NULL)
{
if(p->data%2==0)
sumeven=sumeven+p->data;
else
sumodd=sumodd+p->data;
p=p->llink;
}
cout<<"the sum of even number is \n"<<sumeven<<endl;
```

```
cout<<"the sum of odd number is \n"<<sumodd<<endl;
}
//************************
void rev_list(sll*&f)
{
sll*pre=NULL;
sll*current=NULL;
sll*next=NULL;
current=f;
pre=NULL;
while(current!=NULL)
{
next=current->llink;
current->llink=pre;
pre=current;
current=next;
}
f=pre;
}
//***********************
main()
{
//clrscr();
int x;
sll*f,*n=NULL;
f=intl();
int op;
do{
cout<<"\n enter 1 to addbegin \n 2 to  add  after specific data \n 3 to add end \n
4 to display list \n 5 to exit \n 6 to add befor specific node \n 7 delete from
begin \n 8  delete from end \n 9  delete from midlist \n 10 to orderlist \n 11 to
split list \n 12 to find sum of odd and even number \n 13 to revers list \n 14
delet in end\n";
cin>>op;
switch(op)
{
case 1:
n=creatnode();
cout<<"enter information";
cin>>n->data;
addbeg(f,n);
break;
case 2:
n=creatnode();
cout<<"enter information";
cin>>n->data;
cout<<"enter data you want to add after it \n";
cin>>x;
addaft(f,n,x);
break;
case 3:
n=creatnode();
cout<<"enter information";
cin>>n->data;
addend(f,n);
break;
case 4:
```

```
cout<<"\n @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@\n";
visitlist(f);
break;
case 6:
int y;
n=creatnode();
cout<<"enter information of node";
cin>>n->data;
cout<<"enter data you want to add befor it \n";
cin>>y;
addbef(f,n,y);
break;
case 7:
dbeg(f);
break;
case 8:
dend(f);
break;
case 9:
int k;
cout<<"enter data you want to delete it from midlist";
cin>>k;
dmid(f,k);
break;
case 10:
n=creatnode();
cout<<"enter information";
cin>>n->data;
odev(f,n);
break;
case 11:
int e;
cout<<"enter data you want to split list from it \n";
cin>>e;
div_list(f,e);
break;
case 12:
findsum(f);
break;
case 13:
rev_list(f);
break;
case 14: dlend(f);break;
}
}while(op!=5);
}
```



- اعداد الطالب : عبد الرحمن يحيى محمد صلح
- جامعة صنعاء
- كلية الحاسوب وتكنولوجيا المعلومات
- مستوى ثاني
- قسم نظم معلومات (Information System)

**تم جمع الافكار و الدروس من بعض المواقع ومن عدة كتب والأغلب من محاضرة**

**د/ فضل باعلوي**
**أ /هــــبــــة**
**أ/ مروة الهادي**

**في حالة وجود أي خطأ اتمنى ان تبلغوني في اسرع وقت ممكن**


للتواصل على البريد الإلكتروني
700hnoon@gmail.com
او على الفيس بوك
http://www.facebook.com/hnoon2015
عبد الرحمن يحي صلح